# Table of Contents

## 1 Introduction

1.1 New Features ................................. 14
   1.1.1 Version 2022 ................................. 14
   1.1.2 Version 2021 ................................. 14
   1.1.3 Version 2020 ................................. 15
   1.1.4 Version 2019 ................................. 15
   1.1.5 Version 2018 ................................. 16

1.2 Basic Concepts ................................. 17

1.3 Security Concepts ................................. 19

1.4 How It Works ................................. 21

1.5 Web Administration Interface ................................. 23

1.6 Logging on to FlowForce Server ................................. 25

## 2 Setting Up FlowForce Server

2.1 Setup on Windows ................................. 28
   2.1.1 Installing on Windows ................................. 28
   2.1.2 Installing on Windows Server Core ................................. 29
   2.1.3 Installing LicenseServer (Windows) ................................. 33
   2.1.4 Network and Service Configuration (Windows) ................................. 34
   2.1.5 Licensing FlowForce Server (Windows) ................................. 35

2.2 Setup on Linux ................................. 39
   2.2.1 Installing on Linux ................................. 39
   2.2.2 Installing LicenseServer (Linux) ................................. 40
   2.2.3 Licensing FlowForce Server (Linux) ................................. 41

2.3 Setup on macOS ................................. 45
   2.3.1 Installing on macOS ................................. 45
   2.3.2 Installing LicenseServer (macOS) ................................. 46
   2.3.3 Licensing FlowForce Server (macOS) ................................. 47

2.4 Upgrading FlowForce Server ................................. 50
3 Configure the Server

3.1 Important Paths ................................................................. 54
3.2 Setup Page ........................................................................ 55
3.3 Defining the Network Settings ........................................... 60
   3.3.1 Configuration File Reference .................................... 63
3.4 Setting up SSL Encryption .................................................. 69
   3.4.1 Signing SSL Certificates with a Certificate Authority .... 69
   3.4.2 Creating Self-Signed SSL Certificates ....................... 72
   3.4.3 Private Key Requirements ........................................ 81
   3.4.4 Enabling SSL for FlowForce Web Server ................. 81
   3.4.5 Enabling SSL for FlowForce Server ....................... 83
   3.4.6 Enabling SSL between FlowForce Web Server and FlowForce Server ................................. 85
3.5 Setting the Default Time Zone ............................................. 88
3.6 Setting Mail Parameters ..................................................... 89
3.7 Directory Service Settings .................................................. 90
3.8 Logging Settings ............................................................... 93
3.9 Starting and Stopping Services (Linux) ............................... 99
3.10 Starting and Stopping Services (macOS) ............................. 100
3.11 Starting and Stopping Services (Windows) ......................... 101
3.12 FlowForce Server Application Data .................................. 102
3.13 Localizing FlowForce Server ............................................ 104
3.14 Backup, Data Recovery and Migration ............................... 105
   3.14.1 Backup .................................................................... 105
   3.14.2 Data Recovery and Migration .................................. 106

4 Manage User Access ......................................................... 110

4.1 Users and Roles ............................................................... 111
   4.1.1 Create Users ............................................................ 111
   4.1.2 Create Roles ........................................................... 112
   4.1.3 Import Domain Users and Roles ................................. 112
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.4</td>
<td>Default Users and Roles</td>
<td>114</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Rename Users and Roles</td>
<td>115</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Assign Roles to Users</td>
<td>115</td>
</tr>
<tr>
<td>4.1.7</td>
<td>Assign Roles to Other Roles</td>
<td>116</td>
</tr>
<tr>
<td>4.1.8</td>
<td>Reset the Root Password</td>
<td>116</td>
</tr>
<tr>
<td>4.2</td>
<td>Privileges</td>
<td>118</td>
</tr>
<tr>
<td>4.2.1</td>
<td>How Privileges Work</td>
<td>118</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Viewing Privilege Reports</td>
<td>121</td>
</tr>
<tr>
<td>4.3</td>
<td>Permissions and Containers</td>
<td>124</td>
</tr>
<tr>
<td>4.3.1</td>
<td>How Permissions Work</td>
<td>124</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Understanding Containers</td>
<td>127</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Creating, Renaming, and Moving Containers</td>
<td>129</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Viewing Container Permissions</td>
<td>131</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Setting Container Permissions</td>
<td>132</td>
</tr>
<tr>
<td>4.3.6</td>
<td>Restricting Access to the /public Container</td>
<td>134</td>
</tr>
<tr>
<td>4.4</td>
<td>Password Policies</td>
<td>136</td>
</tr>
<tr>
<td>4.4.1</td>
<td>How Password Policies Work</td>
<td>136</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Creating and Assigning Password Policies</td>
<td>136</td>
</tr>
<tr>
<td>5</td>
<td>Job Configuration</td>
<td>138</td>
</tr>
<tr>
<td>5.1</td>
<td>Credentials</td>
<td>139</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Define Credentials</td>
<td>139</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Refer to Credentials from Jobs</td>
<td>141</td>
</tr>
<tr>
<td>5.2</td>
<td>Create a Job</td>
<td>143</td>
</tr>
<tr>
<td>5.3</td>
<td>Duplicate a Job</td>
<td>144</td>
</tr>
<tr>
<td>5.4</td>
<td>Input Parameters</td>
<td>145</td>
</tr>
<tr>
<td>5.5</td>
<td>Execution Steps</td>
<td>147</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Add a Step</td>
<td>147</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Process Steps Sequentially</td>
<td>149</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Process Steps Conditionally</td>
<td>150</td>
</tr>
<tr>
<td>5.5.4</td>
<td>Step Errors</td>
<td>154</td>
</tr>
<tr>
<td>5.5.5</td>
<td>Retry on Error</td>
<td>157</td>
</tr>
<tr>
<td>5.5.6</td>
<td>Error Handling Strategies</td>
<td>159</td>
</tr>
<tr>
<td>5.5.7</td>
<td>Postponed Steps</td>
<td>162</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.5.8</td>
<td>Data Types in Steps</td>
<td>168</td>
</tr>
<tr>
<td>5.6</td>
<td>Execution Result</td>
<td>171</td>
</tr>
<tr>
<td>5.7</td>
<td>Triggers</td>
<td>172</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Timer Triggers</td>
<td>173</td>
</tr>
<tr>
<td>5.7.2</td>
<td>File System Triggers</td>
<td>174</td>
</tr>
<tr>
<td>5.7.3</td>
<td>HTTP Triggers</td>
<td>176</td>
</tr>
<tr>
<td>5.8</td>
<td>Jobs as Web Services</td>
<td>178</td>
</tr>
<tr>
<td>5.9</td>
<td>Cache Job Results</td>
<td>182</td>
</tr>
<tr>
<td>5.10</td>
<td>Queue Settings</td>
<td>185</td>
</tr>
<tr>
<td>5.11</td>
<td>Windows Network Paths</td>
<td>188</td>
</tr>
<tr>
<td>5.12</td>
<td>FTP Support</td>
<td>189</td>
</tr>
<tr>
<td>6</td>
<td>Import/Export Configuration Data</td>
<td>190</td>
</tr>
<tr>
<td>6.1</td>
<td>Exporting Configuration Data</td>
<td>191</td>
</tr>
<tr>
<td>6.2</td>
<td>Including or Excluding Sensitive Data</td>
<td>196</td>
</tr>
<tr>
<td>6.3</td>
<td>Importing Configuration Data</td>
<td>198</td>
</tr>
<tr>
<td>6.4</td>
<td>Handling Missing Dependencies</td>
<td>199</td>
</tr>
<tr>
<td>7</td>
<td>Monitor Job Execution</td>
<td>201</td>
</tr>
<tr>
<td>7.1</td>
<td>Job Statuses</td>
<td>204</td>
</tr>
<tr>
<td>7.2</td>
<td>Stop Jobs</td>
<td>207</td>
</tr>
<tr>
<td>7.3</td>
<td>Log View</td>
<td>209</td>
</tr>
<tr>
<td>7.4</td>
<td>Instance Log</td>
<td>211</td>
</tr>
<tr>
<td>8</td>
<td>FlowForce Expressions</td>
<td>213</td>
</tr>
<tr>
<td>8.1</td>
<td>Computing an Expression</td>
<td>214</td>
</tr>
<tr>
<td>8.2</td>
<td>The Expression Language Rules</td>
<td>215</td>
</tr>
<tr>
<td>8.3</td>
<td>Embedding Expressions in String Fields</td>
<td>219</td>
</tr>
<tr>
<td>8.4</td>
<td>Calling Expression Functions</td>
<td>220</td>
</tr>
<tr>
<td>8.5</td>
<td>FlowForce Data Types</td>
<td>221</td>
</tr>
<tr>
<td>8.6</td>
<td>Operators</td>
<td>224</td>
</tr>
</tbody>
</table>
9 Integration with Altova Product Line

9.1 Preparing Files for Server Execution

9.2 Deploying Mappings to FlowForce Server

9.3 Running Mappings and Transformations as Jobs
   9.3.1 Credentials in Mapping Functions
   9.3.2 Dynamic Authentication
   9.3.3 Resources

9.4 Accessing the Mapping/Transformation Result

9.5 Integration with RaptorXML Server

9.6 Tool Files

10 Command Line Interface

10.1 help

10.2 assignlicense

10.3 compactdb

10.4 createdb

10.5 debug

10.6 exportresourcestrings

10.7 foreground

10.8 initdb

10.9 install

10.10 licenseserver

10.11 migratedb

10.12 repair

10.13 resetpassword

10.14 setdeflang (sdl)

10.15 start

10.16 uninstall

10.17 upgradedb

10.18 verifylicense

11 Built-in Functions
11.1 /system......................................................................................................................... 280
  11.1.1 abort ....................................................................................................................... 280
  11.1.2 compute.................................................................................................................. 281
  11.1.3 compute-string....................................................................................................... 283
  11.1.4 create-file............................................................................................................... 284

11.2 /system/filesystem........................................................................................................ 286
  11.2.1 copy ......................................................................................................................... 286
  11.2.2 delete....................................................................................................................... 287
  11.2.3 mkdir....................................................................................................................... 288
  11.2.4 move......................................................................................................................... 289
  11.2.5 rmdir......................................................................................................................... 290

11.3 /system/ftp................................................................................................................... 291
  11.3.1 delete....................................................................................................................... 292
  11.3.2 delete-wildcard...................................................................................................... 295
  11.3.3 mkdir....................................................................................................................... 298
  11.3.4 move......................................................................................................................... 300
  11.3.5 list.............................................................................................................................. 304
  11.3.6 retrieve.................................................................................................................... 307
  11.3.7 retrieve-wildcard................................................................................................. 310
  11.3.8 rmdir......................................................................................................................... 313
  11.3.9 store.......................................................................................................................... 316
  11.3.10 store-wildcard....................................................................................................... 321

11.4 /system/mail.................................................................................................................. 325
  11.4.1 send......................................................................................................................... 325
  11.4.2 send-mime.............................................................................................................. 326

11.5 /system/maintenance.................................................................................................... 331
  11.5.1 archive-log............................................................................................................. 331
  11.5.2 cleanup-files......................................................................................................... 331
  11.5.3 truncate-log........................................................................................................... 332

11.6 /system/shell.................................................................................................................. 333
  11.6.1 commandline......................................................................................................... 333

12 Expression Functions 335
12.1 Step Result Functions ................................................................................................. 336
<table>
<thead>
<tr>
<th>Section</th>
<th>Functions</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1.1</td>
<td>error-message</td>
<td>336</td>
</tr>
<tr>
<td>12.1.2</td>
<td>exitcode</td>
<td>337</td>
</tr>
<tr>
<td>12.1.3</td>
<td>failed-step</td>
<td>338</td>
</tr>
<tr>
<td>12.1.4</td>
<td>results</td>
<td>340</td>
</tr>
<tr>
<td>12.1.5</td>
<td>retry-count</td>
<td>342</td>
</tr>
<tr>
<td>12.1.6</td>
<td>stdout</td>
<td>343</td>
</tr>
<tr>
<td>12.1.7</td>
<td>stderr</td>
<td>344</td>
</tr>
<tr>
<td>12.2</td>
<td>Stream Functions</td>
<td>345</td>
</tr>
<tr>
<td>12.2.1</td>
<td>as-file</td>
<td>345</td>
</tr>
<tr>
<td>12.2.2</td>
<td>content</td>
<td>346</td>
</tr>
<tr>
<td>12.2.3</td>
<td>empty-stream</td>
<td>346</td>
</tr>
<tr>
<td>12.2.4</td>
<td>stream-from-string</td>
<td>346</td>
</tr>
<tr>
<td>12.2.5</td>
<td>stream-open</td>
<td>347</td>
</tr>
<tr>
<td>12.3</td>
<td>File System Functions</td>
<td>349</td>
</tr>
<tr>
<td>12.3.1</td>
<td>list-files</td>
<td>349</td>
</tr>
<tr>
<td>12.3.2</td>
<td>list-directories</td>
<td>349</td>
</tr>
<tr>
<td>12.3.3</td>
<td>read-lines</td>
<td>350</td>
</tr>
<tr>
<td>12.4</td>
<td>File Path Functions</td>
<td>352</td>
</tr>
<tr>
<td>12.4.1</td>
<td>extension</td>
<td>352</td>
</tr>
<tr>
<td>12.4.2</td>
<td>filename</td>
<td>352</td>
</tr>
<tr>
<td>12.4.3</td>
<td>filename-with-extension</td>
<td>353</td>
</tr>
<tr>
<td>12.4.4</td>
<td>join-paths</td>
<td>353</td>
</tr>
<tr>
<td>12.4.5</td>
<td>parent-directory</td>
<td>354</td>
</tr>
<tr>
<td>12.5</td>
<td>List Functions</td>
<td>357</td>
</tr>
<tr>
<td>12.5.1</td>
<td>char</td>
<td>357</td>
</tr>
<tr>
<td>12.5.2</td>
<td>code</td>
<td>358</td>
</tr>
<tr>
<td>12.5.3</td>
<td>from-to</td>
<td>359</td>
</tr>
<tr>
<td>12.5.4</td>
<td>join</td>
<td>359</td>
</tr>
<tr>
<td>12.5.5</td>
<td>length</td>
<td>361</td>
</tr>
<tr>
<td>12.5.6</td>
<td>list</td>
<td>362</td>
</tr>
<tr>
<td>12.5.7</td>
<td>nth</td>
<td>362</td>
</tr>
<tr>
<td>12.5.8</td>
<td>slice</td>
<td>363</td>
</tr>
<tr>
<td>12.6</td>
<td>String Functions</td>
<td>364</td>
</tr>
<tr>
<td>12.6.1</td>
<td>concat</td>
<td>364</td>
</tr>
<tr>
<td>12.6.2</td>
<td>contains</td>
<td>364</td>
</tr>
</tbody>
</table>
## 12.6.3 `ends-with` ................................. 365
## 12.6.4 `find-all` .................................. 366
## 12.6.5 `number` .................................... 366
## 12.6.6 `split` ....................................... 367
## 12.6.7 `starts-with` .............................. 367
## 12.6.8 `string` ...................................... 368
## 12.6.9 `string-join` .............................. 368
## 12.6.10 `string-length` ......................... 369
## 12.6.11 `substring` .............................. 369
## 12.6.12 `trim` ...................................... 370
## 12.6.13 `trim-start` ............................. 371
## 12.6.14 `trim-end` ............................... 371

### 12.7 Boolean Functions .......................... 372

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>372</td>
</tr>
<tr>
<td><code>any</code></td>
<td>372</td>
</tr>
<tr>
<td><code>false</code></td>
<td>373</td>
</tr>
<tr>
<td><code>if</code></td>
<td>373</td>
</tr>
<tr>
<td><code>not</code></td>
<td>374</td>
</tr>
<tr>
<td><code>true</code></td>
<td>374</td>
</tr>
</tbody>
</table>

### 12.8 Runtime Information Functions ........ 375

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>instance-id</code></td>
<td>375</td>
</tr>
<tr>
<td><code>log</code></td>
<td>375</td>
</tr>
<tr>
<td><code>slot-number</code></td>
<td>377</td>
</tr>
</tbody>
</table>

### 12.9 MIME Expression Functions ............ 378

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add-mime-header</code></td>
<td>378</td>
</tr>
<tr>
<td><code>add-mime-headers</code></td>
<td>379</td>
</tr>
<tr>
<td><code>current-message-id</code></td>
<td>379</td>
</tr>
<tr>
<td><code>get-mime-content-disposition-param</code></td>
<td>380</td>
</tr>
<tr>
<td><code>get-mime-content-id</code></td>
<td>381</td>
</tr>
<tr>
<td><code>get-mime-content-type-param</code></td>
<td>381</td>
</tr>
<tr>
<td><code>get-mime-header</code></td>
<td>382</td>
</tr>
<tr>
<td><code>get-mime-headers</code></td>
<td>383</td>
</tr>
<tr>
<td><code>get-stream-filename</code></td>
<td>383</td>
</tr>
<tr>
<td><code>is-file</code></td>
<td>384</td>
</tr>
<tr>
<td><code>is-mime-content-type</code></td>
<td>384</td>
</tr>
</tbody>
</table>
12.9.12 mime-content-encode ................................................................. 385
12.9.13 mime-flatten ................................................................. 386
12.9.14 mime-multipart ................................................................. 386
12.9.15 mime-multipart-related ................................................................. 387
12.9.16 mime-multipart-from-list ................................................................. 388
12.9.17 mime-parse ................................................................. 388
12.9.18 mime-split-multipart ................................................................. 389
12.9.19 new-message-id ................................................................. 389
12.9.20 reset-mime-headers ................................................................. 390
12.9.21 set-mime-content-disposition ................................................................. 390
12.9.22 set-mime-content-id ................................................................. 391
12.9.23 set-mime-header ................................................................. 391
12.9.24 set-mime-headers ................................................................. 392

13 Job Configuration Examples ................................................................. 393
13.1 Create a "Hello, World!" Job ................................................................. 395
13.2 Check if a Path Exists ................................................................. 398
13.3 Copy Files ................................................................. 402
13.4 Create a Job from a MapForce Mapping ................................................................. 407
13.5 Use a Job as Step of Another Job ................................................................. 415
13.6 Create a Directory Polling Job ................................................................. 418
13.7 Add Error Handling to a Job ................................................................. 424
13.8 Expose a Job as a Web Service ................................................................. 430
13.9 Post JSON to FlowForce Web Service ................................................................. 438
13.10 Cache Job Results ................................................................. 446
13.11 Create a Job from a StyleVision Transformation ................................................................. 450
13.12 Validate a Document with RaptorXML ................................................................. 458
13.13 Validate XML with Error Logging ................................................................. 460
13.14 Run XSLT with RaptorXML ................................................................. 465
13.15 Generate PDFs from XML Files ................................................................. 470

14 Glossary ................................................................. 482
14.1 C ................................................................. 483
1 Introduction

FlowForce Server is a cross-platform software solution used to automate tasks on Windows, Linux, and macOS servers and workstations through a Web interface.

FlowForce Server integrates with other Altova server products (MapForce Server, StyleVision Server, and both flavors of Raptor XML Server) and extends their functionality by means of recurring or on-demand jobs, including jobs that run as Web services. For example, by virtue of integration with MapForce Server and StyleVision Server, you can run a MapForce mapping or a StyleVision transformation as a recurring FlowForce job. Likewise, by virtue of integration with RaptorXML Server, you can validate XML or JSON files as an on-demand job exposed as a Web service.

With FlowForce Server you can also create and automate various other common server tasks, such as sending emails, managing files on the local system or network, managing files through a File Transfer Protocol (FTP), running shell scripts, and others.

Last updated: 19 October 2021
1.1 New Features

This section describes new features of each FlowForce Server release. For more details, see the respective subsection.

1.1.1 Version 2022

Version 2022

- The Running Jobs section of the Home page now displays the following job execution details: all jobs, recently finished, starting, and running jobs.
- A new system function called create-file has been introduced. This function allows you to store stream content in a file you would like to keep for future use.

1.1.2 Version 2021

Version 2021 Release 2

- The existing FlowForce built-in functions from the /system/ftp library now support options for connecting to a server via FTPS (FTP via SSL).
- New FTP functions are available that enable uploading, retrieving, and deleting files on a remote FTP server using wildcards. Specifically, if you connect to the FTP server through FTP or FTPS, you can use the functions delete-wildcard, retrieve-wildcard, and store-wildcard from the /system/ftp library.
- When creating a file system trigger, you can set the minimum polling interval to 1 second (previously, the minimum interval was 30 seconds).
- Statistics Detail Page: changes in the color scheme and labeling.
- File Path Functions: join-paths is a new function that allows combining paths supplied as arguments into one path.

Version 2021

- The Log View page has been optimized to load records faster and includes new navigation and filtering options, as well as the ability to save the current state of the log as a permanent link.
- A new Log Instance page is available that is dedicated exclusively to viewing one logged job instance at a time. From this page, you can export the logged information to a .zip archive in order to view it later or send it to another party. You can also load previously exported job instances into the "Log Instance" page and view them for post mortem debugging, for example.
- There are new Logging Settings available that let you configure whether certain logging details should be stored or skipped for logging purposes. You can also configure the level of logging detail based on the job outcome. For example, on job failure, you might want to keep full tracing information in the log, whereas on successful execution you might want to keep only the most basic information.
- You can configure certain logging settings not only at application level, but also for specific FlowForce Server jobs. See Logging rules at object level.
1.1.3 Version 2020

Version 2020 Release 2

- It is possible to retry the execution of one or more steps multiple times in case of error, see Retry on Error.
- A job can execute steps in a postponed way, after returning the result, which is particularly suitable in case of jobs invoked through Web service calls, see Postponed Steps.
- A new optional Host name field is available in the setup page, see Defining the Network Settings. This makes SSL configuration more flexible, and also enables you to test run Web services directly from the job configuration page.
- It is possible to configure file system triggers to fire when new files or directories are added to a specified directory. This trigger is different from the existing "Modified date" in that it does not fire if files within the polled directory are subsequently modified.
- The procedure for accessing the Setup Page has been simplified.

Version 2020

- You can define credentials both in MapForce and FlowForce Server, and either embed them into the mapping at design time, or supply them as parameters to the execution step in FlowForce Server, see Credentials in Mapping Functions.
- When defining a credential object, you can restrict it to a specific domain of usage. "Usage" can be one or more of the following: job execution, FTP, HTTP.
- Portable file, folder, and database references defined in MapForce (also known as "Global Resources") can be deployed to FlowForce Server and be consumed by a mapping function. If necessary, you can change directly in FlowForce the resources (file, folder, or database references) used by a mapping function—this will affect all FlowForce jobs using that function. You can also create or edit resources directly in FlowForce Server, with some limitations, see Resources.
- When exporting job configuration data to another FlowForce Server instance or to a .zip archive, you can optionally choose to export sensitive data as well, see Importing and Exporting Configuration Data.

1.1.4 Version 2019

Version 2019 Release 3

- Web services created with FlowForce Server can now accept the body of the HTTP POST request as a job parameter, see Web Service Parameters. For an example, see Post JSON to FlowForce Web Service.
- The logging capabilities of FlowForce Server have been enhanced, with the help of the new log expression function and new logging settings, see Changing the Logging Settings.
Version 2019

- FlowForce Server can now be integrated not only with Windows Active Directory, but also with other Directory Service providers that support LDAP (Lightweight Directory Access Protocol), see Changing the Directory Service Settings.
- A new FlowForce Server built-in function is available, `send-mime`, that enables you to customize e-mail messages sent by FlowForce Server (for example, prepare the HTML body or attachments by running a StyleVision Server transformation). The new function also makes it possible to customize the SMTP headers using MIME Expression Functions available in FlowForce.

1.1.5 Version 2018

Version 2018 Release 2

- (Windows specific) Enhancement to Windows Active Directory integration. If the Windows machine where FlowForce Server runs is part of any domains, these domains now become automatically available for selection in the Settings page (see Changing the Directory Service Settings).

Version 2018

- Internal updates and optimizations
1.2 Basic Concepts

Jobs
A job is a core concept in FlowForce Server. It represents a task or a sequence of tasks to be executed by the server. Jobs can be as simple as one-step tasks such as sending an email. However, you can also create jobs that perform multiple actions and pass the result (for example, a file) as parameter to another job. A job consists of input parameters, steps, triggers, and other settings.

Structure of a FlowForce job

Input parameters
In the context of a FlowForce Server job, input parameters represent some arbitrary information supplied to the job based on which the job will execute in a particular way. Job input parameters are similar to function arguments in a programming language. Input parameters can be of various types (for example, file or directory references, text, numbers, Boolean values, and others). For some job types, FlowForce Server creates the input parameters automatically.
Triggers
When you create a job, you must specify conditions (or criteria) that will trigger the job. These conditions are known as triggers. FlowForce Server monitors any defined triggers and executes the job whenever the trigger condition is met.

Steps
Steps define what the FlowForce Server job must actually do (for example, delete a file, execute a MapForce mapping, or send an email). In its simplest form, a step is just an operation with either failed or successful outcome, and it requires a function to be executed (see Execution step). However, there are also steps that provide means to execute other steps conditionally or in a loop (see Choose step, Error/success Handling step, and For-Each step). You can create as many steps as required within the same job, and you can set the order in which they must take place.

Functions
In the context of a job, a function is an instruction understood by FlowForce Server that performs some operation on the target file system. A function can be one of the following:

- A built-in FlowForce function (see Built-in Functions)
- A StyleVision transformation
- A MapForce mapping
- The execution step of a job

Most functions have input parameters. Any mandatory input parameters must be supplied by the caller in order for the step to execute successfully.

Execution result
In FlowForce Server, a step's execution result defines what is returned after the step is executed (for example, a file, or some text). When working with jobs, you can explicitly declare a step's execution result to be of a specific data type (such as String or Boolean), or be discarded. You typically need to declare the data type of the execution result if you intend to use it in other jobs, or if you want to cache the result.

Credentials
A credential object stores authentication information. This is typically the combination of user name and password associated with a user account on the operating system where the FlowForce Server job runs, but it can also be a set of HTTP or FTP credentials.

Settings
When creating a job, you can configure the following optional settings:

- Make the job available as a Web service (see also Exposing Jobs as Web Services).
- Limit the number of instances running in parallel for the same job (see also Defining Queue Settings).
- Cache the result returned by the job (see also Caching Job Results).
1.3 Security Concepts

FlowForce Server uses a role-based user access control mechanism configurable according to the needs and structure of your organization or business model. For example, you can organize and package jobs and credentials into special data containers that require access rights in order to be viewed or modified. Only users with corresponding access rights would then be able to access data inside the container.

Containers

As the name of the term implies, a container is data packaged together. In FlowForce Server, containers can be roughly compared to folders on an operating system. Containers can contain any of the following: jobs, credentials, functions, and other containers. By setting permissions on containers, you can control who can view or access the data inside them. Organizing data into containers and setting up the relevant permissions for each container is a good security practice.

Users

Users are persons who log on to FlowForce Server to configure jobs, deploy MapForce or StyleVision transformations, or manage the FlowForce Server. The actions available to users in FlowForce Server depend on the following:
   a) Their assigned permissions or privileges
   b) The permissions and privileges assigned to any roles that users are members of.

Roles

Roles are named sets of privileges that help enforce security based on the business need. The typical role-based security involves at least two roles: an administrator and a standard user. Each role is defined by the privileges granted to that role. For example, administrators can change their own password and that of other users, whereas standard users can change only their own password. You can assign roles to users and revoke roles from users as necessary.

Privileges

Privileges define what users can do in FlowForce Server (for example, set own password, read users and roles, stop any job, and so on). Privileges are different from permissions in the sense that permissions control user access to containers, whereas privileges are effective globally across FlowForce Server. The following simple rule might help you distinguish quickly between privileges and permissions: privileges are global, permissions are local.

Like permissions, privileges can be assigned both to individual users and to roles. Therefore, when users log on to FlowForce Server, their set of effective privileges is determined by:
   a) the privileges they have been assigned directly
   b) the privileges assigned to any roles that the user is member of.

Permissions

Permissions control user access to containers. Like privileges, permissions can be granted both to users and to roles. Therefore, if a user is a part of a role, any permissions granted to the role will automatically apply to the user as well.

By default, permissions set on a container are inherited from the parent container. For example, let’s assume
that container A has a child container B. Users who have permission to access container A will have by default permission to access container B as well. However, an administrator can redefine the permissions of any user or role at every level of the container hierarchy.

**Password policies**
FlowForce Servers uses password policies to help administrators manage the complexity of user passwords. A password policy is a set of minimum requirements that a user password must meet in order to be valid (for example, at least $N$ characters long).
1.4 How It Works

Altova website: Workflow automation tool

The typical FlowForce Server installation is illustrated in the following diagram. Components that are optionally licensed are represented with dashed borders.

Typical FlowForce Server installation

As the diagram shows, the typical FlowForce Server installation consists of several server products (or, in technical terms, services) that are installed on the same server machine as FlowForce Server. The FlowForce Server solution itself is composed of two services: FlowForce Web Server and FlowForce Server. These two run as separate services and can be configured, started, or stopped separately. The manner in which these two services are managed depends on the operating system family (Linux, Windows, macOS), as further described in this documentation.

FlowForce Web Server accepts and validates requests from clients, and passes them to FlowForce Server. FlowForce Server is the core of the FlowForce Server solution and runs as a background service without a graphical user interface. FlowForce Server continuously checks for trigger conditions, starts and monitors job execution, and writes detailed logs. In addition to this, FlowForce Server listens to requests for jobs that were exposed as Web services (it can be configured to accept HTTP requests both from the local machine and from remote clients, see Defining the Network Settings).

FlowForce Web Server, on the other hand, handles requests to the Web administration interface where you define or monitor jobs, or manage various FlowForce settings. FlowForce Web Server accepts HTTP (or HTTPS) connections from the following types of clients:
<table>
<thead>
<tr>
<th><strong>Web browser</strong></th>
<th>The Web browser is used to configure FlowForce Server jobs and other settings (for an overview, see <a href="https://www.altova.com/documentation.html">Web Administration Interface</a>).</th>
</tr>
</thead>
</table>
| **MapForce Enterprise or Professional Edition** | MapForce is a data mapping desktop application where you visually design the mappings that transform your data or convert it from one format to another.  

Once the mappings are created and tested in MapForce, you can deploy them to FlowForce Server, in order to convert them into flexibly configurable jobs. For example, you can configure the mapping jobs to run at a specific time daily, or whenever a file is added to a monitored directory.  

In order to run jobs created from MapForce mappings, FlowForce Server calls MapForce Server (or MapForce Server Advanced Edition), whose role is to actually execute the mappings and produce the resulting output files.  

Both MapForce Server and MapForce Server Advanced Edition integrate seamlessly with FlowForce; however, only one of them can be installed at the same time alongside FlowForce. By default, when installing FlowForce, you will be prompted to optionally install the MapForce Server Advanced Edition. |
| **StyleVision Enterprise or Professional Edition** | StyleVision is a desktop application used to design reports and forms based on XML, SQL database, and XBRL inputs.  

Once a stylesheet has been tested and debugged, it can be deployed to FlowForce Server. The deployed files are then available for use in any transformation job on the server.  

To execute jobs created from deployed StyleVision transformations, FlowForce Server calls StyleVision Server, whose role is to execute the transformation and produce the resulting output files. |

For further information about each product, refer to the Altova documentation page ([https://www.altova.com/documentation.html](https://www.altova.com/documentation.html)).

**RaptorXML Server**

Altova RaptorXML Server (also called RaptorXML for short) is Altova's third-generation, super-fast XML processor, optimized for the latest standards and parallel computing environments. Designed to be highly cross-platform capable, the engine takes advantage of today's ubiquitous multi-core computers to deliver lightning-fast processing of XML. RaptorXML is available in two editions: (i) **RaptorXML Server** and (ii) **RaptorXML+XBRL Server**. The **RaptorXML+XBRL Server** edition includes support for validating and processing XBRL (eXtensible Business Reporting Language) documents, in addition to XML.

When RaptorXML Server is installed on the same server as FlowForce Server, its functions become available as built-in FlowForce Server functions. This means that you can create jobs that validate or check the well-formedness of XML documents, or transform XSLT and XQuery documents. For more information, see [Integration with RaptorXML Server](https://www.altova.com/documentation.html).
1.5 Web Administration Interface

The FlowForce Server Web administration interface allows you to administer the server and configure jobs. You can access the Web administration interface from a Web browser at the configured address and port.

The following pages are available in the Web administration interface:

- **Home**
  - Shows the list of running jobs and the list of active timers.

- **Configuration**
  - Displays the currently defined FlowForce containers, jobs, credentials, and functions. To view the contents and further information about any object, click the corresponding record.
  - The following containers are available by default:
    - `/public`
    - `/system`
    - `/RaptorXML (if you have licensed RaptorXML Server)`
  - For more information about containers, see Understanding Containers. From the Configuration page, you can also manage containers, jobs, credentials, and functions, and set permissions on containers if you have the relevant access rights.

- **Log**
  - Opens the Log View page that shows log entries, including both server-related and job-related messages.

- **Administration**
  - The Administration page enables you to perform actions related to server configuration and user management.

- **Users**
  - Allows you to manage user access.

- **Roles**
  - Allows you to create, delete, and manage roles. For more information, see Users and Roles.
Password policies
 Allows you to establish password complexity rules.

Reports
 Allows you to view reports on currently assigned user privileges.

Settings
 Allows you to define the default time zone, mail server, and settings that let you integrate FlowForce Server with Active Directory or an LDAP-compliant server. For more information, see Configuring the Server.

Help
 Opens the FlowForce Server documentation in a separate browser tab or window.
1.6 Logging on to FlowForce Server

To manage FlowForce Server (create jobs, add users, and so on), you must log on to the Web Administration Interface at the configured HTTP(S) address and port (for example, http://localhost:8082). For information about configuring this URL, see Defining the Network Settings (look for the settings grouped under "FlowForce Web Server").

By default, after a fresh installation of FlowForce Server, you can log on with the username root and password root.

For security reasons, make sure to change the default root password immediately after first login to FlowForce Server.

If authentication with a Directory Service provider (such as Active Directory) is configured, domain users can also log on to FlowForce. In this case, the login page includes an additional drop-down list where you can select the domain. To use standard HTTP authentication instead of Directory Service authentication, select Directly from the Login drop-down list.
Clients which access Web services exposed by FlowForce Server (typically, at a URL like http://localhost:4646/service/SomeService) may also use Active Directory authentication as an alternative to HTTP authentication. For Active Directory authentication to be possible, the username must be prefixed with NT/ and must include the domain name, for example NT/john.doe@my.domain.com, see also Exposing Jobs as Web Services.

For information about how to configure Windows domain authentication, see Changing the Directory Service Settings.
2 Setting Up FlowForce Server

This section describes installation, licensing and other setup procedures. It is organized into the following sections:

- Setting Up FlowForce Server
- Setup on Windows
- Setup on Linux
- Setup on macOS
- Upgrading FlowForce Server
- Post-Licensing Tasks
- Migrating FlowForce Server to a New Machine
2.1 Setup on Windows

This section describes the installation and licensing of FlowForce Server on Windows systems.

System requirements
- Windows 7 SP1 with Platform Update, Windows 8, Windows 10, Windows 11
- Windows Server 2008 R2 SP1 with Platform Update or newer

Prerequisites
- Perform installation as a user with administrative privileges.
- From version 2021 onwards, a 32-bit version of FlowForce Server cannot be installed over a 64-bit version, or a 64-bit version over a 32-bit version. You must either (i) remove the older version before installing the newer version or (ii) upgrade to a newer version that is the same bit version as your older installation.

2.1.1 Installing on Windows

FlowForce Server is available for installation on Windows systems. The broad installation and setup procedure is described below. For detailed information about specific parts of the installation procedure, see their respective topics.

Installing FlowForce Server
To install FlowForce Server, download the installation package from the Altova Download Center (http://www.altova.com/download.html), run it and follow the on-screen instructions. You can select your installation language from the box in the lower left area of the wizard. Note that this selection also sets the default language of FlowForce Server. You can change the language later from the command line.

After installation, the FlowForce Server executable will be located by default at:

<ProgramFilesFolder>\Altova\FlowForceServer2022\bin\FlowForceServer.exe

Uninstalling FlowForce Server
Uninstall FlowForce Server as follows:

1. Right-click the Windows Start button and select Settings.
2. Open the Control Panel (start typing "Control Panel" and click the suggested entry).
3. Under Programs, click Uninstall a program.

Trial license
During the installation process, you will be given the option of requesting a 30-day trial license for FlowForce Server. After submitting the request, a trial license will be sent to the email address you registered.
## 2.1.2 Installing on Windows Server Core

Windows Server Core is a minimal Windows installation that does not use a number of GUI features. You can install FlowForce Server on a Windows Server Core machine as follows:

1. Download the FlowForce Server installer executable from the Altova website. This file is named `FlowForceServer.exe`. Make sure to choose the executable matching your server platform (32-bit or 64-bit).
2. On a standard Windows machine (not the Windows Server Core machine), run the command `FlowForceServer.exe /u`. This unpacks the `.msi` file to the same folder as the installer executable.
3. Copy the unpacked `.msi` file to the Windows Server Core machine.
4. If you are updating an earlier version of FlowForce Server, shut down FlowForce Server before carrying out the next step.
5. Use the `.msi` file for the installation by running the command `msiexec /i FlowForceServer.msi`. This starts the installation on Windows Server Core.

**Note:** When upgrading to a major version, you can retain your FlowForce Server settings by using the properties listed in the subsections of this section: (i) Webserver Properties, (ii) SSL-Webserver Properties, and (iii) Service Properties.

### Important: Keep the MSI file!

Note the following points:

- Keep the extracted `.msi` file in a safe place. You will need it later to uninstall, repair, or modify your installation.
- If you want to rename the MSI file, do this before you install FlowForce Server.
- The MSI filename is stored in the registry. You can update its name there if the filename has changed.

#### Register FlowForce Server with LicenseServer

If you are installing FlowForce Server for the first time or are upgrading to a major version, you will need to register FlowForce Server with an Altova LicenseServer on your network. If you are upgrading to a non-major version of FlowForce Server, then the previous LicenseServer registration will be known to the installation and there is no need to register FlowForce Server with LicenseServer. However, if you want to change the LicenseServer that is used by FlowForce Server at any time, then you will need to register FlowForce Server with the new LicenseServer.

To register FlowForce Server with an Altova LicenseServer during installation, run the installation command with the `REGISTER_WITH_LICENSE_SERVER` property, as listed below, providing the name or address of the LicenseServer machine as the value of the property, for example:

```
msiexec /i FlowForceServer.msi REGISTER_WITH_LICENSE_SERVER="localhost"
```

To register FlowForce Server with an Altova LicenseServer after installation, run the following command:

```
msiexec /r FlowForceServer.msi REGISTER_WITH_LICENSE_SERVER="<MyLS-IPAddress>"
```
Useful commands

Given below are a set of commands that are useful in the installation context.

To test the return value of the installation, run a script similar to that below. The return code will be in the `%errorlevel%` environment variable. A return code of 0 indicates success.

```
start /wait msiexec /i FlowForceServer.msi /q
echo %errorlevel%
```

For a silent installation with a return code and a log of the installation process:

```
start /wait msiexec /i FlowForceServer.msi /q /L*v! <pathToInstallLogFile>
```

To modify the installation:

```
msiexec /m FlowForceServer.msi
```

To repair the installation:

```
msiexec /r FlowForceServer.msi
```

To uninstall FlowForce Server:

```
msiexec /x FlowForceServer.msi
```

To uninstall FlowForce Server silently and report the detailed outcome in a log file:

```
start /wait msiexec /x FlowForceServer.msi /q /L*v! <pathToUninstallLogFile>
```

To install FlowForce Server using another language (available language codes are: German=de; Spanish=es; French=fr):

```
msiexec /i FlowForceServer.msi INSTALLER_LANGUAGE=<languageCode>
```

Note: On Windows Server Core, the charts functionality of FlowForce Server will not be available.

2.1.2.1 Webserver Properties

You can configure the FlowForce Server web server by using the properties given below. To set a property, run the installation command with the property setting appended, like this:

```
msiexec /i FlowForceServer.msi FF_WebServer_Host=127.0.0.1
```

List of properties

*FlowForce Server*

Properties of the FlowForce Server web server:

**FF_WebServer_Host=<IP4 Address>**

Use 127.0.0.1 if you want to access the web server from this machine only. Use 0.0.0.0 to make the web server accessible globally.

**FF_WebServer_Port=<Port Number>**

Specifies the port that is used to access the web server.
FF_WebServer_Enabled=<0 or 1>
Select 1 to enable listening at the currently set port. Select 0 to disable listening at this port.

FlowForce Server Web
Properties of the FlowForce Server Web interface web server:

FFWeb_WebServer_Host=<IP4 Address>
Use 127.0.0.1 if you want to access the web server from this machine only. Use 0.0.0.0 to make the web server accessible globally.

FFWeb_WebServer_Port=<Port Number>
Specifies the port that is used to access the web server.

FFWeb_WebServer_Enabled=<0 or 1>
Select 1 to enable listening at the currently set port. Select 0 to disable listening at this port.

2.1.2.2 SSL-Webserver Properties
You can configure the FlowForce Server SSL web server by using the properties given below. To set a property, run the installation command with the property setting appended, like this:

    msiexec /i FlowForceServer.msi FF_SSLWebServer_Host=127.0.0.1

List of properties

FlowForce Server
To configure the FlowForce Server SSL web server, use the following properties:

FF_SSLWebServer_Host=<IP4 Address>
Use 127.0.0.1 if you want to access the SSL web server (for encrypted transmission) from this machine only. Use 0.0.0.0 to make the SSL web server accessible globally.

FF_SSLWebServer_Port=<Port Number>
Specifies the port that is used to access the SSL web server (for encrypted transmission).

FF_SSLWebServer_Enabled=<0 or 1>
Select 1 to enable listening at the currently set port. Select 0 to disable listening at this port.

FF_SSLWebServer_Certificate=<Path-to-certificate-file>
Full path to a SSL certificate, enclosed in double-quotes.

FF_SSLWebServer_PrivateKey=<Path-to-private-key-file>
Full path to a private key file, enclosed in double-quotes.

FlowForce Server Web
To configure the SSL web server of the FlowForce Server Web interface, use the following properties:
FFWeb_SSLWebServer_Host=<IP4 Address>
Use 127.0.0.1 if you want to access the SSL web server from this machine only. Use 0.0.0.0 to make the SSL web server accessible globally.

FFWeb_SSLWebServer_Port=<Port Number>
Specifies the port that is used to access the SSL web server.

FFWeb_SSLWebServer_Enabled=<0 or 1>
Select 1 to enable listening at the currently set port. Select 0 to disable listening at this port.

2.1.2.3 Service Properties
You can configure the FlowForce Server service by using the properties given below. To set a property, run the installation command with the property setting appended, like this:

    msiexec /i FlowForceServer.msi FF_Service_DisplayName=FlowForceServer

List of properties
FlowForce service
To configure FlowForce Server services, use the following properties:

FF_Service_DisplayName=<Service Display Name>
Name that will be displayed for the service. Enclose the name in double quotes.

FF_Service_StartType=<Startup Type>
Specifies how the service is started during a system start-up. Values can be one of: auto | auto-
delayed | demand | disabled.

FF_Service_Username=<UserName>
Specifies the log-on user for the service. Use one of: LocalSystem | NT Authority\LocalService | NT
Authority\NetworkService | <any user with relevant rights>.

FF_Service_Password=<Password>
The password of the service's start user in plain text.(Hint: Use the installer's user interface to avoid
entering plain text passwords.) No password is required if the user name is any of: LocalSystem | NT
Authority\LocalService | NT Authority\NetworkService.

FlowForce Web service
To configure FlowForce ServerWeb services, use the following properties:

FFWeb_Service_DisplayName=<Service Display Name>
Name that will be displayed for the service. Enclose the name in double quotes.

FFWeb_Service_StartType=<Startup Type>
Specifies how the service is started during a system start-up. Values can be one of: auto | auto-
delayed | demand | disabled.

FFWeb_Service_Username=<UserName>
Specifies the log-on user for the service. Use one of: LocalSystem | NT Authority\LocalService | NT Authority\NetworkService | <any user with relevant rights>.

**FFWeb_Service_Password=<Password>**

The password of the service's start user in plain text. (Hint: Use the installer's user interface to avoid entering plain text passwords.) No password is required if the user name is any of: LocalSystem | NT Authority\LocalService | NT Authority\NetworkService.

### 2.1.3 Installing LicenseServer (Windows)

In order for FlowForce Server to work, it must be licensed via an [Altova LicenseServer](https://www.altova.com/license-server) on your network. When you install FlowForce Server on Windows systems, you can install LicenseServer together with FlowForce Server. If a LicenseServer is already installed on your network, you do not need to install another one—unless a newer version of LicenseServer is required. (See next point, LicenseServer versions.)

During the installation process of FlowForce Server, check or uncheck the option for installing LicenseServer as appropriate. Note the following points:

- If you have not installed LicenseServer yet, leave the default settings as is. The wizard will install the latest version on the computer where you are running the wizard.
- If you have not installed LicenseServer yet and want to install Altova LicenseServer on another computer, clear the check box *Install Altova LicenseServer on this machine* and choose *Register Later*. In this case, you will need to install LicenseServer separately and register FlowForce Server afterwards.
- If LicenseServer has already been installed on your computer but is a lower version than the one indicated by the installation wizard, leave the default settings as is. In this case, the installation wizard will automatically upgrade your LicenseServer version. The existing registration and licensing information will be carried over to the new version of LicenseServer.
- If LicenseServer has already been installed on your computer or network and has the same version as the one indicated by the wizard, do the following:
  - Clear the check box *Install Altova LicenseServer on this machine*.
  - Under *Register this product with*, choose the LicenseServer with which you want to register FlowForce Server. Alternatively, choose *Register Later*. Note that you can always select *Register Later* if you want to ignore the LicenseServer associations and carry on with the installation of FlowForce Server.

For information about how to register and license FlowForce Server with Altova LicenseServer, see the section [Licensing FlowForce Server](https://www.altova.com/license-server).

### LicenseServer versions

- Altova server products must be licensed either with the version of LicenseServer that is appropriate to the installed FlowForce Server version or with a later version of LicenseServer.
- The LicenseServer version that is appropriate for a particular version of FlowForce Server is displayed during the installation of FlowForce Server. You can install this version of LicenseServer along with FlowForce Server. Alternatively, you can install LicenseServer separately.
- Before installing a newer version of LicenseServer, any older one must be de-installed. The LicenseServer installer will do this automatically if it detects an older version.
- LicenseServer versions are backwards compatible. They will work with older versions of FlowForce...
Setting Up FlowForce Server Setup on Windows

- If you install a new version of FlowForce Server and if your installed LicenseServer version is older than the appropriate LicenseServer, install the latest version of LicenseServer available on the Altova website.
- At the time of LicenseServer de-installation, all registration and licensing information held in the older version of LicenseServer will be saved to a database on your server machine. This data will be imported automatically into the newer version when the newer version is installed.
- The version number of the currently installed LicenseServer is given at the bottom of the LicenseServer configuration page (all tabs).

Current version: 3.7

2.1.4 Network and Service Configuration (Windows)

During the installation of FlowForce Server, you can configure settings for accessing FlowForce Server via the network and for running FlowForce Server as a Windows service (screenshot below). You can configure FlowForce Server and FlowForce Web Server separately by selecting their respective tabs. If you have chosen to install RaptorXML Server together with FlowForce Server, then you can configure settings for RaptorXML Server separately by selecting its tab.

The settings listed below are available. Leave the default settings as they are if they are acceptable to you or if you are not sure about them. If you wish to change a setting, select its Change button (see screenshot above).

- The port to use for unencrypted communication with FlowForce Server.
- Whether secure (SSL-encrypted) connections to FlowForce Server are allowed. If yes, then on which port. By default, secure connections are disabled. For more information, see the section about setting up SSL encryption.
- Windows service settings. These include:
  - The way FlowForce Server should start as a Windows service: automatic, on demand, delayed automatic, or disabled.
  - The user account to be used by FlowForce Server for the Windows service: Local System, Local Service, Network Service, or Other User. If you select Other User, you can set the username and password of this user, similar to how this is done in the Windows Services management console. Note that the selected user must have read/write access to c:\ProgramData\Altova. Otherwise, the installation or startup could fail. Note that if you select Other User to run the services, you must ensure the following privileges have been granted: Adjust memory quotas for a process (SeIncreaseQuotaPrivilege) and Replace a process level token.
To grant these privileges, go to the Windows applet Local Security Policy and open the subtree Local Policies/User Rights Assignment. Local Security Policy can be accessed via the Start menu search box. These privileges are only required for the main FlowForce service, not for FlowForceWeb. Importantly, the file access rights are required for both.

You can change the settings after installation. For information about modifying network configuration, see Defining the Network Settings. To modify the Windows service configuration, open the Windows Services management console (by typing Services.msc in a command line window) and change the required service from there.

2.1.5 Licensing FlowForce Server (Windows)

In order to use FlowForce Server, it must be licensed with Altova LicenseServer. Licensing is a two-step process:

1. **Register FlowForce Server** with LicenseServer. Registration is done from FlowForce Server.
2. **Assign a license** to FlowForce Server from LicenseServer. Download the latest version of LicenseServer from the Altova website, and install it on your local machine or a machine on your network.

These steps are described in this section. For detailed information, see the LicenseServer user manual at the Altova website.

2.1.5.1 Start LicenseServer, FlowForce Server

Altova LicenseServer (LicenseServer for short) and FlowForce Server are both started via Altova ServiceController.

Altova ServiceController

Altova ServiceController (ServiceController for short) is an application for conveniently starting, stopping and configuring Altova services on Windows systems. ServiceController is installed with Altova LicenseServer and with Altova server products that are installed as services (DiffDog Server, FlowForce Server, Mobile Together Server, and RaptorXML(+XBRL) Server). ServiceController can be accessed via the system tray (screenshot below).

To specify that ServiceController starts automatically on logging in to the system, click the ServiceController icon in the system tray to display the ServiceController menu (screenshot below), and then toggle on the command Run Altova ServiceController at Startup. (This command is toggled on by default.) To exit ServiceController, click the ServiceController icon in the system tray and, in the menu that appears (see screenshot below), click Exit Altova ServiceController.
Start LicenseServer
To start LicenseServer, click the ServiceController icon in the system tray, hover over Altova LicenseServer in the menu that pops up (see screenshot below), and then select Start Service from the LicenseServer submenu. If LicenseServer is already running, then the Start Service option will be disabled. You can also stop the service via ServiceController.

Start FlowForce Server
To start FlowForce Server, click the ServiceController icon in the system tray, hover over Altova FlowForce Server in the menu that pops up, and then select Start Service from the FlowForce Server submenu. If FlowForce Server is already running, the Start Service option will be disabled. You can also stop the service via ServiceController.

2.1.5.2 Register FlowForce Server
In order to be able to license FlowForce Server from Altova LicenseServer, FlowForce Server must be registered with LicenseServer
To register FlowForce Server from the command line interface, use the `licenseserver` command and supply the address of the LicenseServer machine.

```
FlowForceServer licenseserver [options] ServerName-Or-IP-Address
```

For example, if `localhost` is the name of the server on which LicenseServer is installed:

```
FlowForceServer licenseserver localhost
```

After successful registration, go to the Client Management tab of LicenseServer's configuration page to assign a license to FlowForce Server.

**Note:** For more information about registering Altova products with LicenseServer, see the LicenseServer user manual.

### 2.1.5.3 License FlowForce Server

After successfully registering FlowForce Server, it will be listed in the Client Management tab of the configuration page of LicenseServer. Go there and assign a license to FlowForce Server.

The licensing of Altova server products is based on the number of processor cores available on the product machine. For example, a dual-core processor has two cores, a quad-core processor four cores, a hexa-core processor six cores, and so on. The number of cores licensed for a product must be greater than or equal to the number of cores available on that server machine, whether the server is a physical or virtual machine. For example, if a server has eight cores (an octa-core processor), you must purchase at least one 8-core license. You can also combine licenses to achieve the core count. So, two 4-core licenses can also be used for an octa-core server instead of one 8-core license.

If you are using a computer server with a large number of CPU cores but only have a low volume to process, you may also create a virtual machine that is allocated a smaller number of cores and purchase a license for that number. Such a deployment, of course, would have less processing speed than if all available cores on the server were utilized.

**Note:** Each Altova server product license can be used for only one client machine at a time, even if the license has unused licensing capacity. (A client machine is the machine on which the Altova server product is installed.) For example, if a 10-core license is used for a client machine that has 6 CPU cores, then the remaining 4 cores of licensing capacity cannot be used simultaneously for another client machine.

**FlowForceServer and MapForceServer licensing**

FlowForce Server Advanced Edition and MapForce Server Advanced Edition will run only on machines with two or more cores.

When assessing the number of cores you should license, take into account the data volume you need to process and the processing time your business environment is expected to allow for. In most scenarios, a larger number of cores means more volume of data processed in less time. Given below are a few application-specific tips:

- FlowForce Server runs as a multi-threaded application. If the number of concurrent requests to the server is big, an insufficient number of cores will lead to latency (waiting times). For example, if you are exposing jobs as Web services, there may be hundreds of concurrent requests from clients. In this case, FlowForce Server will significantly benefit from a larger number of cores.
• MapForce Server will utilize a single core at a time, per mapping. Therefore, if you need to run multiple mappings simultaneously, a larger number of cores is highly recommended. For example, when MapForce Server runs under FlowForce Server management, several mapping jobs may overlap and run concurrently, depending also on the setup. Note, however, that if the volumes processed by your mappings are extremely big, latency could still occur.

In addition to the above, note that there are various external factors that typically influence the processing volumes and times that your server is capable of handling (for example, the hardware, the current load on the CPU, memory allocation of other applications running on the server). In order to get the most accurate performance measurements, it is generally advisable to first run the tools in your environment and expose them to actual factors and data specific to your business.

**Single-thread execution**

If an Altova server product allows single-thread execution, an option for *Single-thread execution* will be available. In these cases, if an Altova server-product license for only one core is available in the license pool, a machine with multiple cores can be assigned this one-core license. In such a case, the machine will run that product on a single core. Processing will therefore be slower, because multi-threading (which is possible on multiple cores) will not be available. The product will be executed in single thread mode on that machine.

To assign a single-core license to a multiple-core machine in LicenseServer, select the *Limit to single thread execution* check box for that product.

**Estimate of core requirements**

There are various external factors that influence the data volumes and processing times your server can handle (for example: the hardware, the current load on the CPU, and memory allocation of other applications running on the server). In order to measure performance as accurately as possible, test the applications in your environment with data volumes and in conditions that approximate as closely as possible to real business situations.
2.2 Setup on Linux

This section describes the installation and licensing of FlowForce Server on Linux systems (Debian, Ubuntu, CentOS, RedHat).

System Requirements (Linux)

- CentOS 7 or newer
- RedHat 7 or newer
- Debian 8 or newer
- Ubuntu 16.04 LTS or newer

Prerequisites

- Perform installation either as root user or as a user with sudo privileges.
- The previous version of FlowForce Server must be uninstalled before a new one is installed.
- If you are installing FlowForce Server with other Altova server products, it is recommended that you install FlowForce Server first.
- The following libraries are required as a prerequisite to install and run the application. If the packages below are not already available on your Linux machine, run the command yum (or apt-get if applicable) to install them.

<table>
<thead>
<tr>
<th>Required by</th>
<th>CentOS, RedHat</th>
<th>Debian</th>
<th>Ubuntu</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlowForce Server</td>
<td>libidn, krb5-libs</td>
<td>libidn11, libgssapi-krb5-2</td>
<td>libidn11, libgssapi-krb5-2</td>
</tr>
<tr>
<td>LicenseServer</td>
<td>krb5-libs</td>
<td>libgssapi-krb5-2</td>
<td>libgssapi-krb5-2</td>
</tr>
</tbody>
</table>

2.2.1 Installing on Linux

FlowForce Server is available for installation on Linux systems. Its installation and setup procedure is described below. Perform installation either as root user or as a user with sudo privileges.

Uninstall FlowForce Server

If you need to uninstall a previous version of FlowForce Server, do this as follows. On the Linux command line interface (CLI), you can check which Altova server products are installed with the following command:

[Debian, Ubuntu]: `dpkg --list | grep Altova`

[CentOS, RedHat]: `rpm -qa | grep server`

[CentOS, RedHat]: `rpm -qa | grep flowforce`

Note: The command `rpm -qa | grep flowforce` will only give you server packages for FlowForce, whereas the command `rpm -qa | grep server` may list many unrelated server packages.

If FlowForce Server is not installed, go ahead with the installation as documented below in Install FlowForce Server.
If you need to uninstall an old version of FlowForce Server, do this with the following command:

[Debian, Ubuntu]: `sudo dpkg --remove flowforceserver`

[CentOS, RedHat]: `sudo rpm -e flowforceserver`

On Debian and Ubuntu systems, it might happen that FlowForce Server still appears in the list of installed products after it has been installed. In this case, run the `purge` command to clear FlowForce Server from the list. You can also use the `purge` command instead of the `remove` command listed above.

[Debian, Ubuntu]: `sudo dpkg --purge flowforceserver`

Download the FlowForce Server Linux package

FlowForce Server installation packages for the following Linux systems are available at the Altova website.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Package extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian</td>
<td>.deb</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>.deb</td>
</tr>
<tr>
<td>CentOS</td>
<td>.rpm</td>
</tr>
<tr>
<td>RedHat</td>
<td>.rpm</td>
</tr>
</tbody>
</table>

After downloading the Linux package, copy it to any directory on the Linux system. Since you will need an Altova LicenseServer in order to run FlowForce Server, you may want to download LicenseServer from the Altova website at the same time as you download FlowForce Server, rather than download it at a later time.

Install FlowForce Server

In a terminal window, switch to the directory where you have copied the Linux package. For example, if you copied it to a user directory called MyAltova (that is located, say, in the /home/User directory), then switch to this directory as follows:

```
cd /home/User/MyAltova
```

Install FlowForce Server with the following command:

[Debian]: `sudo dpkg --install flowforceserver-2022-debian.deb`

[Ubuntu]: `sudo dpkg --install flowforceserver-2022-ubuntu.deb`

[CentOS]: `sudo rpm -ivh flowforceserver-2022-1.x86_64.rpm`

[RedHat]: `sudo rpm -ivh flowforceserver-2022-1.x86_64.rpm`

**Note:** You may need to adjust the name of the package above to match the current release or service pack version.

The FlowForce Server package will be installed in the folder:

```
/opt/Altova/FlowForceServer2022
```

### 2.2.2 Installing LicenseServer (Linux)

In order for FlowForce Server to work, it must be licensed via an Altova LicenseServer on your network. On Linux systems, Altova LicenseServer will need to be installed separately. Download LicenseServer from the
Altova website and copy the package to any directory on the Linux system. Install it just like you installed FlowForce Server (see previous topic).

[Debian]: sudo dpkg --install licenseserver-3.7-debian.deb  
[Ubuntu]: sudo dpkg --install licenseserver-3.7-ubuntu.deb  
[CentOS]: sudo rpm -ivh licenseserver-3.7-1.x86_64.rpm  
[RedHat]: sudo rpm -ivh licenseserver-3.7-1.x86_64.rpm

The LicenseServer package will be installed in:

/opt/Altova/LicenseServer

For information about how to register and license FlowForce Server with Altova LicenseServer, see the section Licensing FlowForce Server. Also see the LicenseServer documentation for more detailed information.

LicenseServer versions

- Altova server products must be licensed either with the version of LicenseServer that is appropriate to the installed FlowForce Server version or with a later version of LicenseServer.
- The LicenseServer version that is appropriate for a particular version of FlowForce Server is displayed during the installation of FlowForce Server. You can install this version of LicenseServer along with FlowForce Server. Alternatively, you can install LicenseServer separately.
- Before installing a newer version of LicenseServer, any older one must be de-installed. The LicenseServer installer will do this automatically if it detects an older version.
- LicenseServer versions are backwards compatible. They will work with older versions of FlowForce Server.
- If you install a new version of FlowForce Server and if your installed LicenseServer version is older than the appropriate LicenseServer, install the latest version of LicenseServer available on the Altova website.
- At the time of LicenseServer de-installation, all registration and licensing information held in the older version of LicenseServer will be saved to a database on your server machine. This data will be imported automatically into the newer version when the newer version is installed.
- The version number of the currently installed LicenseServer is given at the bottom of the LicenseServer configuration page (all tabs).

Current version: 3.7

2.2.3 Licensing FlowForce Server (Linux)

In order to use FlowForce Server, it must be licensed with Altova LicenseServer. Licensing is a two-step process:

2. Assign a license to FlowForce Server from LicenseServer. Download the latest version of LicenseServer from the Altova website, and install it on your local machine or a machine on your network.

These steps are described in this section. For detailed information, see the LicenseServer user manual at the Altova website.
2.2.3.1 Start LicenseServer, FlowForce Server

This topic describes how to start Altova LicenseServer (LicenseServer for short) and FlowForce Server. You must have administrator (root) privileges to be able to start these programs, so you should run these commands as the root user. If you are logged in as root, you can leave out the sudo keyword from the commands listed below.

Start LicenseServer

To correctly register and license FlowForce Server with LicenseServer, LicenseServer must be running as a daemon on the network. Start LicenseServer as a daemon with the following command:

```
[< Debian 8] sudo /etc/init.d/licenseserver start
[>= Debian 8] sudo systemctl start licenseserver
[< CentOS 7] sudo initctl start licenseserver
[>= CentOS 7] sudo systemctl start licenseserver
[< Ubuntu 15] sudo initctl start licenseserver
[>= Ubuntu 15] sudo systemctl start licenseserver
[RedHat] sudo initctl start licenseserver
```

If at any time you need to stop LicenseServer, replace start with stop in the above commands. For example:

```
sudo systemctl stop licenseserver
```

Start FlowForce Server

Start FlowForce Server as a daemon with the command appropriate for your system from those listed below. Since FlowForce Server consists of two services, `flowforceserver` and `flowforcewebserver`, start each separately using the appropriate command:

```
[< Debian 8] sudo /etc/init.d/flowforceserver start
[>= Debian 8] sudo systemctl start flowforceserver
[< CentOS 7] sudo initctl start flowforceserver
[>= CentOS 7] sudo systemctl start flowforceserver
[< Ubuntu 15] sudo initctl start flowforceserver
[>= Ubuntu 15] sudo systemctl start flowforceserver
[RedHat] sudo initctl start flowforceserver
```

If at any time you need to stop FlowForce Server, replace start with stop in the above commands. For example:

```
sudo systemctl stop flowforceserver
```
Check status of daemons
To check if a daemon is running, run the following command, replacing `servicename` with the name of the daemon you want to check:

```
sudo service servicename status
```
FlowForce and MapForce Server licensing

FlowForce Server Advanced Edition and MapForce Server Advanced Edition will run only on machines with two or more cores.

When assessing the number of cores you should license, take into account the data volume you need to process and the processing time your business environment is expected to allow for. In most scenarios, a larger number of cores means more volume of data processed in less time. Given below are a few application-specific tips:

- FlowForce Server runs as a multi-threaded application. If the number of concurrent requests to the server is big, an insufficient number of cores will lead to latency (waiting times). For example, if you are exposing jobs as Web services, there may be hundreds of concurrent requests from clients. In this case, FlowForce Server will significantly benefit from a larger number of cores.
- MapForce Server will utilize a single core at a time, per mapping. Therefore, if you need to run multiple mappings simultaneously, a larger number of cores is highly recommended. For example, when MapForce Server runs under FlowForce Server management, several mapping jobs may overlap and run concurrently, depending also on the setup. Note, however, that if the volumes processed by your mappings are extremely big, latency could still occur.

In addition to the above, note that there are various external factors that typically influence the processing volumes and times that your server is capable of handling (for example, the hardware, the current load on the CPU, memory allocation of other applications running on the server). In order to get the most accurate performance measurements, it is generally advisable to first run the tools in your environment and expose them to actual factors and data specific to your business.

Single-thread execution

If an Altova server product allows single-thread execution, an option for Single-thread execution will be available. In these cases, if an Altova server-product license for only one core is available in the license pool, a machine with multiple cores can be assigned this one-core license. In such a case, the machine will run that product on a single core. Processing will therefore be slower, because multi-threading (which is possible on multiple cores) will not be available. The product will be executed in single thread mode on that machine.

To assign a single-core license to a multiple-core machine in LicenseServer, select the Limit to single thread execution check box for that product.

Estimate of core requirements

There are various external factors that influence the data volumes and processing times your server can handle (for example, the hardware, the current load on the CPU, and memory allocation of other applications running on the server). In order to measure performance as accurately as possible, test the applications in your environment with data volumes and in conditions that approximate as closely as possible to real business situations.
2.3 Setup on macOS

This section describes the installation and licensing of FlowForce Server on macOS systems.

System Requirements (Linux)
- macOS 10.13 or newer

Prerequisites
- Ensure that Altova LicenseServer has been installed and is running
- Perform installation either as root user or as a user with sudo privileges.
- The previous version of FlowForce Server must be uninstalled before a new one is installed.
- If you are installing FlowForce Server with other Altova server products, it is recommended that you install FlowForce Server first.
- The macOS machine must be configured so that its name resolves to an IP address. This means that you must be able to successfully ping the host name from the Terminal using the command ping <hostname>.

2.3.1 Installing on macOS

FlowForce Server is available for installation on macOS systems. Its installation and setup procedure is described below.

Uninstall FlowForce Server
Before uninstalling FlowForce Server, stop the service with the following command:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.FlowForceServer2022.plist
```

To check whether the service has been stopped, open the Activity Monitor in Finder and make sure that FlowForce Server is not in the list. In the Applications folder in Finder, right-click the FlowForce Server icon and select Move to Trash. The application will be moved to Trash. You will, however, still need to remove the application from the /usr folder. Do this with the following command:

```
sudo rm -rf /usr/local/Altova/FlowForceServer2022/
```

If you need to uninstall an old version of Altova LicenseServer, you must first stop it running as a service. Do this with the following command:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

To check whether the service has been stopped, open the Activity Monitor in Finder and make sure that LicenseServer is not in the list. Then proceed to uninstall in the same way as described above for FlowForce Server.

Install FlowForce Server

1. Download the disk image (.dmg) file of FlowForce Server from the Altova website (http://www.altova.com/download.html).
2. Click to open the downloaded disk image (.dmg). This causes the FlowForce Server installer to appear as a new virtual drive on your computer.
3. On the new virtual drive, double-click the installer package (.pkg).
4. Go through the successive steps of the installer wizard. These are self-explanatory and include one step in which you have to agree to the license agreement before being able to proceed. See also Licensing FlowForce Server.
5. To eject the drive after installation, right-click it and select Eject.

The FlowForce Server package will be installed in the folder:

`/usr/local/Altova/FlowForceServer2022` (application binaries)
`/var/Altova/FlowForceServer` (data files: database and logs)

The FlowForce Server server daemon starts automatically after installation and a re-boot of the machine. You can always start FlowForce Server as a daemon with the following command:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.FlowForceServer2022.plist
```

After starting the FlowForce Server server daemon, you can open the Config page of FlowForce Server in order to configure FlowForce Server. Open this page by going to the Applications folder in Finder and double-clicking the FlowForce Server icon there.

See the topic Post-Licensing Tasks for additional details.

### 2.3.2 Installing LicenseServer (macOS)

In order for FlowForce Server to work, it must be licensed via an Altova LicenseServer on your network. The LicenseServer installation package is available on the virtual drive you have mounted in the previous step. To install LicenseServer, double-click the installer package included on the virtual drive and follow the on-screen instructions. You will need to accept the license agreement for installation to proceed.

Altova LicenseServer can also be downloaded and installed separately from the Altova website [Altova LicenseServer](http://www.altova.com/download.html).

The LicenseServer package will be installed in the folder:

`/usr/local/Altova/LicenseServer`

For information about how to register FlowForce Server with Altova LicenseServer and license it, see Licensing on macOS.

LicenseServer versions

- Altova server products must be licensed either with the version of LicenseServer that is appropriate to the installed FlowForce Server version or with a later version of LicenseServer.
- The LicenseServer version that is appropriate for a particular version of FlowForce Server is displayed during the installation of FlowForce Server. You can install this version of LicenseServer along with FlowForce Server. Alternatively, you can install LicenseServer separately.
- Before installing a newer version of LicenseServer, any older one must be de-installed. The LicenseServer installer will do this automatically if it detects an older version.
- LicenseServer versions are backwards compatible. They will work with older versions of FlowForce Server.
If you install a new version of FlowForce Server and if your installed LicenseServer version is older than the appropriate LicenseServer, install the latest version of LicenseServer available on the Altova website.

At the time of LicenseServer de-installation, all registration and licensing information held in the older version of LicenseServer will be saved to a database on your server machine. This data will be imported automatically into the newer version when the newer version is installed.

The version number of the currently installed LicenseServer is given at the bottom of the LicenseServer configuration page (all tabs).

Current version: 3.7

2.3.3 Licensing FlowForce Server (macOS)

In order to use FlowForce Server, it must be licensed with Altova LicenseServer. Licensing is a two-step process:

1. **Register FlowForce Server** with LicenseServer. Registration is done from FlowForce Server.
2. **Assign a license** to FlowForce Server from LicenseServer. Download the latest version of LicenseServer from the Altova website, and install it on your local machine or a machine on your network.

These steps are described in this section. For detailed information, see the LicenseServer user manual at the Altova website.

2.3.3.1 Start LicenseServer, FlowForce Server

This topic describes how to start Altova LicenseServer (LicenseServer for short) and FlowForce Server. You must have administrator (root) privileges to be able to start these programs, so you should run these commands as the root user. If you are logged in as root, you can leave out the sudo keyword from the commands listed below.

### Start LicenseServer

To correctly register and license FlowForce Server with LicenseServer, LicenseServer must be running as a daemon. Start LicenseServer as a daemon with the following command:

```bash
sudo launchctl load /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

If at any time you need to stop LicenseServer, replace load with unload in the above command:

```bash
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

### Start FlowForce Server

FlowForce Server server daemon starts automatically after installation and a re-boot of the machine. You can start FlowForce Server as a daemon with the following command:

```bash
sudo launchctl load /Library/LaunchDaemons/com.altova.FlowForceServer2022.plist
```

If at any time you need to stop FlowForce Server, use:

```bash
sudo launchctl unload /Library/LaunchDaemons/com.altova.FlowForceServer2022.plist
```
2.3.3.2 Register FlowForce Server

To register FlowForce Server from the command line interface, use the licenseserver command:

```
sudo /usr/local/Altova/FlowForceServer2022/bin/FlowForceServer licenseserver [options]
ServerName-Or-IP-Address
```

For example, if `localhost` is the name of the server on which LicenseServer is installed:

```
sudo /usr/local/Altova/FlowForceServer2022/bin/FlowForceServer licenseserver localhost
```

In the command above, `localhost` is the name of the server on which LicenseServer is installed. Notice also that the location of the FlowForce Server executable is:

```
/usr/local/Altova/FlowForceServer2022/bin/
```

You can also register FlowForce Server from the FlowForce Server Setup page. In Applications, double-click the FlowForce Server icon. This opens the FlowForce Server Setup page in the browser, where you can carry out the registration.

If the name of the Mac machine cannot resolve to an IP address (see the prerequisites), the browser opens a page with the following message: "FlowForceWeb does not appear to be available at http://<hostname>:<port>. Please restart it and reload this page." If you see this message, do the following: (i) Click the link given in the message; (ii) In the browser's address bar, replace `<hostname>` with either `localhost` or the IP address of your Mac.

After successful registration, go to the Client Management tab of LicenseServer's configuration page to assign a license to FlowForce Server.

**Note:** For more information about registering Altova products with LicenseServer, see the LicenseServer user manual.

2.3.3.3 License FlowForce Server

After successfully registering FlowForce Server, it will be listed in the Client Management tab of the configuration page of LicenseServer. Go there and assign a license to FlowForce Server.

The licensing of Altova server products is based on the number of processor cores available on the product machine. For example, a dual-core processor has two cores, a quad-core processor four cores, a hexa-core processor six cores, and so on. The number of cores licensed for a product must be greater than or equal to the number of cores available on that server machine, whether the server is a physical or virtual machine. For example, if a server has eight cores (an octa-core processor), you must purchase at least one 8-core license. You can also combine licenses to achieve the core count. So, two 4-core licenses can also be used for an octa-core server instead of one 8-core license.

If you are using a computer server with a large number of CPU cores but only have a low volume to process, you may also create a virtual machine that is allocated a smaller number of cores and purchase a license for that number. Such a deployment, of course, would have less processing speed than if all available cores on the server were utilized.
Note: Each Altova server product license can be used for only one client machine at a time, even if the license has unused licensing capacity. (A client machine is the machine on which the Altova server product is installed.) For example, if a 10-core license is used for a client machine that has 6 CPU cores, then the remaining 4 cores of licensing capacity cannot be used simultaneously for another client machine.

FlowForceServer and MapForceServer licensing
FlowForce Server Advanced Edition and MapForce Server Advanced Edition will run only on machines with two or more cores.

When assessing the number of cores you should license, take into account the data volume you need to process and the processing time your business environment is expected to allow for. In most scenarios, a larger number of cores means more volume of data processed in less time. Given below are a few application-specific tips:

- FlowForce Server runs as a multi-threaded application. If the number of concurrent requests to the server is big, an insufficient number of cores will lead to latency (waiting times). For example, if you are exposing jobs as Web services, there may be hundreds of concurrent requests from clients. In this case, FlowForce Server will significantly benefit from a larger number of cores.
- MapForce Server will utilize a single core at a time, per mapping. Therefore, if you need to run multiple mappings simultaneously, a larger number of cores is highly recommended. For example, when MapForce Server runs under FlowForce Server management, several mapping jobs may overlap and run concurrently, depending also on the setup. Note, however, that if the volumes processed by your mappings are extremely big, latency could still occur.

In addition to the above, note that there are various external factors that typically influence the processing volumes and times that your server is capable of handling (for example, the hardware, the current load on the CPU, memory allocation of other applications running on the server). In order to get the most accurate performance measurements, it is generally advisable to first run the tools in your environment and expose them to actual factors and data specific to your business.

Single-thread execution
If an Altova server product allows single-thread execution, an option for Single-thread execution will be available. In these cases, if an Altova server-product license for only one core is available in the license pool, a machine with multiple cores can be assigned this one-core license. In such a case, the machine will run that product on a single core. Processing will therefore be slower, because multi-threading (which is possible on multiple cores) will not be available. The product will be executed in single thread mode on that machine.

To assign a single-core license to a multiple-core machine in LicenseServer, select the Limit to single thread execution check box for that product.

Estimate of core requirements
There are various external factors that influence the data volumes and processing times your server can handle (for example: the hardware, the current load on the CPU, and memory allocation of other applications running on the server). In order to measure performance as accurately as possible, test the applications in your environment with data volumes and in conditions that approximate as closely as possible to real business situations.
2.4 Upgrading FlowForce Server

When you upgrade to a newer version of FlowForce Server, the license of your previous version will be used automatically for the newer version if, during installation:

- the new version is registered with the same LicenseServer as that with which the previous version of FlowForce Server was registered
- you accept the license agreement of FlowForce Server.

The simplest way to carry over a license from the previous version of FlowForce Server to the newer version is to let the installation process implement the required steps. The relevant steps during the installation process are listed below in the order in which they occur:

1. Let the installer register the new version of FlowForce Server with the LicenseServer that holds the license used by the older version of FlowForce Server.
2. Accept the license agreement of FlowForce Server. (If you do not accept the agreement, the new version will not be installed.)

Note: If you do not register FlowForce Server with the correct LicenseServer during the installation process, you will need to register and license FlowForce Server manually with your alternative LicenseServer.
2.5 Post-Licensing Tasks

After you have completed the installation and licensing of FlowForce Server, carry out the following tasks to finish the setup.

1. Enter setup mode to go to the Setup page, where you can specify various network settings, including the interface and port on which FlowForce Server and FlowForce Web Server should listen. See Setup Page.
2. Configure the network address and port of the Web administration interface. See Defining the Network Settings.
3. Log on to the Web administration interface (by default, this is http://localhost:8082 unless you changed the address and port in the previous step). Here you can carry out various administration tasks, such as changing the default password. The default login name and password is root and root, respectively.
4. If other Altova server products have been installed alongside FlowForce Server (for example, MapForce Server, StyleVision Server, RaptorXML Server), you might want to set environment variables for them. See Setting Environment Variables.
2.6 Migrating FlowForce Server to a New Machine

If you want to migrate FlowForce Server from one machine to another (including across supported platforms), follow the guidelines that are linked to from below.

See the section titled Backup, Data Recovery and Migration.
3  Configure the Server

This chapter provides information about configuring FlowForce Server. This includes configuration that you need to perform immediately after installation, as well as various server maintenance or routine tasks such as starting or stopping services, data backup, and others.

You can manage FlowForce Server and its settings using the approaches listed below.

<table>
<thead>
<tr>
<th>From the Setup page</th>
<th>After a new FlowForce Server installation, the first thing that you typically define is the host name (or IP address) and port where the FlowForce Web Server and FlowForce Server services should listen. For further information about this part, see:</th>
</tr>
</thead>
</table>
|                     | · Setup Page 56  
|                     | · Defining the Network Settings 60  
|                     | · Setting up SSL Encryption 64. |

<table>
<thead>
<tr>
<th>From a Web administration interface</th>
<th>Once the network settings mentioned above are set, you can configure the following in any order:</th>
</tr>
</thead>
</table>
|                                     | · Default Time Zone 86  
|                                     | · Mail Parameters 88  
|                                     | · Directory Service Settings 90  
|                                     | · Logging Settings 93  
|                                     | · User Access 103. |

<table>
<thead>
<tr>
<th>From the command line interface</th>
<th>See Command Line Interface 109.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>By editing configuration files</th>
<th>See FlowForce Server Application Data 102.</th>
</tr>
</thead>
</table>
3.1 Important Paths

After installing FlowForce Server, note the following directories where important files are stored:

- Installation directory (INSTALLDIR)
- Application's data directory (DATADIR)

<table>
<thead>
<tr>
<th></th>
<th><strong>FlowForce Server installation directory (INSTALLDIR)</strong></th>
<th><strong>FlowForce Server application data directory (DATADIR)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linux</strong></td>
<td>/opt/Altova/FlowForceServer2022/</td>
<td>/var/opt/Altova/FlowForceServer2022/data</td>
</tr>
<tr>
<td><strong>macOS</strong></td>
<td>/usr/local/Altova/FlowForceServer2022/</td>
<td>/var/Altova/FlowForceServer2022/data</td>
</tr>
<tr>
<td><strong>Windows</strong></td>
<td>C:\Program Files\Altova\FlowForceServer2022\</td>
<td>C:\ProgramData\Altova\FlowForceServer2022\data</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files (x86)\Altova\FlowForceServer2022\</td>
<td></td>
</tr>
</tbody>
</table>
3.2 Setup Page

The FlowForce Server setup page lets you specify various network settings, including the interface and port on which FlowForce Server and FlowForce Web Server should listen. The easiest way to open the setup page in a browser (or obtain its URL) is by running a setup mode with administrative privileges, as described below. For more advanced configuration, see "Advanced setup options" below.

For reference to all the settings that you can configure from the setup page, see Defining the Network Settings.

Linux

To enter the setup mode on Linux:

1. Open a terminal and change to the application's data directory:

   cd /var/opt/Altova/FlowForceServer2022

2. Do one of the following:

   a. If you run Linux with a graphical user interface, run the FlowForce Web server executable with the setup command:

   sudo /opt/Altova/FlowForceServer2022/bin/flowforcewebserver setup

   b. If you run Linux without a graphical user interface, run the same setup command as above, while also adding the --listen option. The latter specifies the interface and port where the setup page should be available. Once this is done, you can connect to the setup page from a browser on a different machine.

Once you perform the steps above, the terminal displays two alternative URLs for the setup page that you can copy-paste into your browser's address bar. In the event that the first URL does not work, use the second one.

macOS

To enter the setup mode on macOS:

- In Applications, double-click the FlowForce Server 2022 icon.

In emergencies, you can also enter the setup mode as follows:

1. Open a terminal with root privileges and change to the application's data directory:

   cd /var/Applications/FlowForceServer2022

2. Run the FlowForce Web server executable with the setup command:

   sudo /usr/local/Applications/FlowForceServer2022/bin/FlowForceWebServer setup
Once you perform the steps above, the terminal displays two alternative URLs for the setup page that you can copy-paste into your browser’s address bar. In the event that the first URL does not work, use the second one.

Windows
To enter the setup mode on Windows:


In emergencies, you can also enter the setup mode as follows:

1. Open a Command Prompt window as administrator and change to the application’s data directory:

   ```
   cd C:\ProgramData\Altova\FlowForceServer2022
   ```

2. Run the FlowForce Web server executable with the setup command:

   ```
   "C:\Program Files\Altova\FlowForceServer2022\bin\FlowForceWebServer.exe" setup
   ```

Setup page
Once you perform the steps above, the setup page opens in a new browser window (or its URL is displayed at the terminal so that you can paste it into a browser window).
On Windows, an informative Command Prompt window also opens, for example:
This window remains open for the duration of the setup and will normally close automatically after you click the **Apply settings and restart FlowForce Server services** button on the setup page.

When you click the **Apply settings and restart FlowForce services** button from the setup page, your configuration is saved to .ini files. More specifically, if you opened the setup page with administrator privileges as described above, the following ini files will be updated:

- DATADIR\flowforcewebserver.ini
- DATADIR\flowforceserver.ini

Where **DATADIR** refers to the following directory:

- /var/opt/Altova/FlowForceServer2022/data (Linux)
- /var/Altova/FlowForceServer2022/data (macOS)
- C:\ProgramData\Altova\FlowForceServer2022\data (Windows)

If you need more flexibility, you can run the `setup` command with more advanced options, as described below.

**Advanced setup options**

The `setup` command supports a few advanced options listed below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| --datadir | Use this option to supply the path to the directory where the .ini configuration files will be written (normally, the **DATADIR** mentioned above). If the target directory requires elevated "write" privileges, you must run the setup mode with a privileged (root or administrator) account that can write to that directory.  

If you do not specify this option, your .ini files will be saved to the \data subdirectory of the current directory. Therefore, you should either run the setup command from the application data directory (as described above), or supply the --datadir option. |
| --listen | By default, every time when you run the setup, the URL of the setup page is regenerated on a free random port, for example: [http://localhost:50492/setup](http://localhost:50492/setup).  

This option lets you specify an alternative interface/port combination to listen to (other than **localhost** or **127.0.0.1**). This is typically useful if you want to access the setup page from a browser on a different machine. |
For example, the command:

```bash
flowforcewebserver setup --listen=0.0.0.0:10008
```

would make the setup listen on port 10008 on all interfaces. Note the following:

- It is not recommended to make the setup run privileged with the actual data directory and binding it to an external network interface. If you intend to do that, the next option (`--key`) is useful. As an alternative, supply a temporary data directory using the `--datadir` option; this prevents the public configuration page from updating the actual FlowForce Server configuration.
- Do not use the same port as the normal (non-setup) FlowForce Web Server or FlowForce Server instance, because when they run that port will be in use already.

If the binding address (interface) is non-local, you may need to configure the operating system's firewall so as to enable access through the designated port.

<table>
<thead>
<tr>
<th>--key</th>
</tr>
</thead>
<tbody>
<tr>
<td>This option enables you to set an access key for the setup page. In this case, it is possible to save the setup page only if the correct access key was provided in the URL. The key can be some arbitrary string which has to be included in the URL. For example, if you run a Linux command like</td>
</tr>
</tbody>
</table>

```bash
flowforcewebserver setup --listen=wild.berries.com:8015
--key=all_cats_love_fish
--datadir=/var/opt/Altova/FlowForceServer2022/data
```

then the URL to connect to is

```
http://wild.berries.com:8015/setup?key=all_cats_love_fish
```

Make sure that the key phrase is secure enough for your purposes.

Note that the setup page does not use HTTPS because it is used itself to configure the HTTPS parameters. Remember that the setup page is not available continuously, but only for the duration of the setup operation.
3.3 Defining the Network Settings

You can specify the host and port on which FlowForce Server and FlowForce Web Server should listen, as well as other network-related settings, from the Setup page. The Setup Page can be opened in various ways, depending on the operating system. Alternatively, most of these settings can be defined by means of configuration files, see Configuration File Reference. The settings defined in the Setup page will be preserved when you install a new minor version of FlowForce Server. If you install a major version, the settings will be preserved only if you opted to migrate data from the previous major version during installation.

The settings you can configure are listed below.

LicenseServer
FlowForce Server must be registered with LicenseServer (see Altova LicenseServer). If you haven't specified a LicenseServer host during installation, enter here the address or host name of the machine where Altova LicenseServer runs. This can be either the address of the local machine (if LicenseServer is installed locally), or a network address.

FlowForce Web Server
This group of settings is applicable to the FlowForce Web Server service, that is, the service responsible for handling HTTP(S) requests from a browser to the FlowForce Server Web administration interface. (To understand the difference between “FlowForce Web Server” and “FlowForce Server”, see How It Works.)
### FlowForce Web Server

#### Unencrypted Connection

<table>
<thead>
<tr>
<th>Enabled</th>
<th>Bind address</th>
<th>Host name</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>Local only 127.0.0.1</td>
<td></td>
<td>8082</td>
</tr>
</tbody>
</table>

#### SSL Encrypted Connection

<table>
<thead>
<tr>
<th>Enabled</th>
<th>Bind address</th>
<th>Host name</th>
<th>Certificate file</th>
<th>Private Key file</th>
<th>Certificate Chain file</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>All interfaces (0.0.0.0)</td>
<td>example.name.com</td>
<td>C:\secure\flowforceweb.crt</td>
<td>C:\secure\flowforceweb.key</td>
<td>C:\secure\intermediate.pem</td>
</tr>
</tbody>
</table>

### Unencrypted connection / Enabled
Select this check box to enable plain HTTP (unencrypted) connections to FlowForce Web Server. By default, plain HTTP (unencrypted) connections are enabled, unless you modified this during installation* or from configuration files, see [Configuration File Reference](#).

* Modifying the network settings during FlowForce Server installation is supported on Windows only.

### Bind address
On Windows, the FlowForce Web Server administration interface is available by default on all network interfaces on port **8082**. On Linux and Mac OS, the port number is chosen randomly during installation. To specify a custom address other than "Local only" or "All interfaces", enter it in the **Other** text box.

If the binding address (interface) is non-local, you may need to configure the operating system's firewall so as to enable access through the designated port.

### Port
Specifies the TCP port on which FlowForce Web Server should listen. The port must not be already in use.

### Host name
The **Host name** field, if non-empty, sets a fixed host name that is used for the binding. It sets the name of the machine running FlowForce Web Server that other machines on the network would use to connect to it.

Normally, FlowForce detects automatically the appropriate host name to use. If
Configure the Server Defining the Network Settings

| SSL Encrypted Connection | See Enabling SSL for FlowForce Web Server. |

**FlowForce Server**

This group of settings is applicable to the *FlowForce Server* service, that is, the service responsible for exposing Web services created from FlowForce jobs to HTTP(S) clients.

### FlowForce Server

**Unencrypted Connection**

- **Enabled**: [ ]
- **Bind address**: Local only (127.0.0.1) ▼
- **Host name**: 
- **Port**: 4646

### SSL Encrypted Connection

- **Enabled**: 
- **Bind address**: All interfaces (0.0.0.0) ▼
- **Host name**: example.name.com
- **Certificate file**: C:\secure\flowforce.crt
- **Private Key file**: C:\secure\flowforce.key
- **Certificate Chain file**: C:\secure\intermediate.pem
- **Port**: 4647

**Unencrypted connection / Enabled**

Select this check box to enable plain HTTP (unencrypted) connections to FlowForce Server. By default, plain HTTP (unencrypted) connections are enabled, unless you modified this during installation* or from configuration files, see Configuration File Reference. *

* Modifying the network settings during FlowForce Server installation is supported on Windows only.
### 3.3.1 Configuration File Reference

The network settings of both FlowForce Server and FlowForce Web Server can be configured either from the Setup Page or by editing .ini configuration files, as described below.

There are two .ini files, one for FlowForce Server (flowforce.ini), and another one for FlowForce Web Server (flowforceweb.ini). The .ini configuration files can be found at the following path:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>/var/opt/Altova/FlowForceServer2022/data/flowforce.ini</td>
</tr>
</tbody>
</table>
Configure the Server

Defining the Network Settings

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Path Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>macOS</td>
<td>/var/opt/Altova/FlowForceServer2022/data/flowforceweb.ini</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\ProgramData\Altova\FlowForceServer2022\data\flowforce.ini C:\ProgramData\Altova\FlowForceServer2022\data\flowforceweb.ini</td>
</tr>
</tbody>
</table>

In the directory above, you can also find two sample .ini files that contain comments and can be used as a template:

- flowforce.ini.template
- flowforceweb.ini.template

After editing the .ini files, remember to restart the corresponding service (FlowForce Server or FlowForce Web Server). For more information, see:

- Starting and Stopping Services (Linux)
- Starting and Stopping Services (macOS)
- Starting and Stopping Services (Windows)

A sample `flowforce.ini` file looks as follows:

```ini
[Listen]
active=1
host=127.0.0.1
port=4646
hostname=

[ListenSSL]
active=1
SSL=1
host=0.0.0.0
port=4647
hostname=

[SSL]
certificate=/path/to/certificate.crt
private_key=/path/to/private_key.key
certificate_chain=/path/to/certificate_chain
```

A sample `flowforceweb.ini` file looks as follows:

```ini
[Listen]
active=1
host=0.0.0.0
port=8082
hostname=example.domain.org

[ListenSSL]
```
The .ini files are organized into sections, as described below. Differences between both files are mentioned below where applicable.

**[Listen]**

| active | (Optional) Activates or deactivates this [Listen] section. Valid values:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>disabled</td>
</tr>
<tr>
<td>1</td>
<td>enabled</td>
</tr>
</tbody>
</table>

For example, active=1 means that HTTP connections are enabled.

| ssl     | (Optional) Enables SSL support for this [Listen] section. Valid values:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>disabled</td>
</tr>
<tr>
<td>1</td>
<td>enabled</td>
</tr>
</tbody>
</table>

To enable SSL support, set ssl=1 and also create a [SSL] section, as shown below.

<table>
<thead>
<tr>
<th>host</th>
<th>Specifies the network bind address of FlowForce (Web) Server, for example, 127.0.0.1. This can be an IPv4 or IPv6 address. Use 0.0.0.0 to listen on all interfaces. For local access only, use 127.0.0.1.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>port</th>
<th>Specifies the port on which FlowForce (Web) Server will listen. Make sure that this port is not already in use.</th>
</tr>
</thead>
</table>

If the binding address (interface) is non-local, you may need to configure the operating system’s firewall so as to enable access through the...
### designate port.

| hostname | The **Host name** field, if non-empty, sets a fixed host name that is used for the binding. It sets the name of the machine running FlowForce Web Server that other machines on the network would use to connect to it.

Normally, FlowForce detects automatically the appropriate host name to use. If you set this field explicitly, then automatic detection will be overridden. You may need to use a value like `somehost` or `somehost.example.org`, depending on the network configuration in your organization.

The host name associated with a binding is used for SSL (see Enabling SSL for FlowForce Web Server) and by Altova Service Controller on Windows. If SSL is enabled, the host name has to match the **Common Name** property of the certificate. |

| max_request_body_size | This option enables you to specify the maximum size, in bytes, of HTTP requests to either FlowForce Server or FlowForce Web Server, for example:

```ini
max_request_body_size=500000000
```

The default, implicit limit is around 100 MB (100,000,000 bytes). You may need to set this option explicitly in the following situations:

1. If you call FlowForce Web services exposed as jobs and the HTTP request body is larger than the default limit.
2. If you deploy mappings from MapForce to FlowForce Server and the input files are larger than the default limit.

For case 1 above, the option must be set only in the `flowforce.ini` file. For case 2, the option must be set in both `flowforce.ini` and `flowforceweb.ini` files. |

### [SSL]

This section defines the SSL/HTTPS connection settings.

| certificate | Specifies the absolute path to the certificate file in PEM format. |
| private_key | Specifies the absolute path to the private key file. |
| certificate_chain | (optional) The path to the certificate chain file. |

### [FlowForce]

This section is applicable only for FlowForce Web Server (the `flowforceweb.ini` file). It defines the connection details between FlowForce Web Server and FlowForce Server.
### ss1
(Optional) Enables SSL support for the connection between FlowForce Web Server and FlowForce Server. Valid values:
- 0 disabled
- 1 enabled

### host
Specifies the IP address or host name of FlowForce Server.

If FlowForce Server is not bound to all interfaces, this value must be the same as the one in the “[Listen]” section of the `flowforce.ini`, otherwise it is 127.0.0.1.

If SSL is enabled, this value must match the “Common Name” property of the certificate configured in “[SSL]” section of the `flowforce.ini`.

### port
Specifies the TCP port on which FlowForce Web Server is to connect to FlowForce Server.

This value must be the same as the one in corresponding “[Listen]” or “[ListenSSL]” section of the `flowforce.ini` that has the same port number.

If SSL is enabled on this port, host and hostname (or just host if hostname is not present) must match the “Common Name” property of the certificate configured in “[SSL]” section of `flowforce.ini`.

### hostname
If non-empty, this field sets a fixed host name that is used by other machines on the network to connect to FlowForce jobs exposed as Web services (see [Exposing Jobs as Web Services](#)).

You may need to use a value like `somehost` or `somehost.example.org`, depending on the network configuration in your organization.

If SSL is enabled on that port and this value is present, this value has to match the “Common Name” property of the certificate configured in the “[SSL]” section of `flowforce.ini`.

The host name is also used by Altova Service Controller. If you don't set the host name, FlowForce detects automatically the first appropriate host name to be used by Altova Service Controller.

If hostname is configured, the FlowForce web interface can present clickable links to navigate to jobs exposed as Web services, including links in the [Active Triggers and Services](#) section of the home page. Also, a Call Web Service button becomes available in the “Service” section of the job configuration page that enables you to call the Web service in a new browser window.

### certificate
(Optional) Defines what server certificate will be accepted by FlowForce Server. If no certificate is given, the system root CA certificates will be used to verify the server certificate. If present, this value must match the certificate that FlowForce Server is using (the one in the `flowforce.ini` file).
[FlowForceWeb]
This section is applicable only for FlowForce Web Server (the `flowforceweb.ini` file).

<table>
<thead>
<tr>
<th>timezone</th>
<th>Specifies the default time zone of FlowForce Web Server, for example</th>
</tr>
</thead>
<tbody>
<tr>
<td>timezone=Europe/Berlin</td>
<td></td>
</tr>
</tbody>
</table>


3.4 Setting up SSL Encryption

You can configure FlowForce so that the following HTTP connections are encrypted with SSL certificates:

1. The connection between a browser and FlowForce Web Server.
2. The connection between a Web service consumer (for example, some client application) and the FlowForce Server service.
3. The internal connection between FlowForce Web Server and FlowForce Server. (For information about how FlowForce Server is different from FlowForce Web Server, see How It Works.)

For connections 1 and 2 above, you need an SSL certificate and a private key corresponding to that certificate. For security reasons, you might want to use a separate SSL certificate and private key for each connection. If you want to use the same certificate and private key for both connections, this requires that both FlowForce Server and FlowForce Web Server have the same fully qualified domain name (FQDN). For example, if FlowForce Web Server listens on `https://somehost:8083`, then FlowForce Server should listen on `https://somehost:4647`. Note that you can always change the port later, only the host name is important in this case.

For connection 3 above, there is no need for a third certificate and private key pair—you can use the same SSL certificate as for FlowForce Server—in this case, FlowForce Web Server acts as HTTP client to FlowForce Server.

To obtain the certificates required to encrypt SSL connections in FlowForce Server, you have the following options:

1. Generate a CSR (Certificate Signing Request) and then have it signed by a public certificate authority (CA), such as DigiCert, Comodo, and others. The vast majority of browsers will trust server certificates signed by such a CA, because the browser (or the operating system) already trusts the CA. For instructions about how to obtain certificates signed by a public certificate authority, see Signing SSL Certificates with a Certificate Authority.

2. Alternatively, if FlowForce Server runs on a private network, and if you have the entitlement to do this in your organization, it is possible to configure your own SSL root certification authority. No browser or operating system trusts such an authority by default, so you will need to configure each machine (or browser, depending on the case) that connects to FlowForce Server to trust your self-signed root certificate. Otherwise, the browser will still display warnings such as “This site is not secure” or the Web service call will not be successful. For more information, see Creating Self-Signed SSL Certificates.

3.4.1 Signing SSL Certificates with a Certificate Authority

Before you can purchase SSL certificates from a trusted certificate authority (CA), you need a private key and a CSR (Certificate Signing Request). The private key must be stored securely and not disclosed to anyone; the CSR will be required by the certificate authority during the ordering process.

You can create the private key and the CSR using a tool that may already exist on your operating system (such as Keychain Access on Mac, openssl on Linux), or third party tools. This example makes use of the OpenSSL toolkit (https://www.openssl.org/). Note that OpenSSL is an open source library, and may need to be compiled before you can use it at the command line. The compilation and installation instructions for OpenSSL vary for each operating system and are outside of the scope of this documentation. On a Linux and Mac machine, it is likely that OpenSSL is already available; otherwise, you can install it or update it from the...
command line. You can quickly check if OpenSSL present by typing the command below (it displays the current OpenSSL version):

```
openssl version
```

On Windows, you can either compile binaries from the official OpenSSL source code, or, alternatively, download a binary distribution that includes OpenSSL. See also [https://www.openssl.org/community/binaries.html](https://www.openssl.org/community/binaries.html).

**To obtain a signed SSL certificate:**

1. Create the private key. The following OpenSSL command generates a key called `flowforce.key` that is 2048-bit in size (the minimum encryption strength normally accepted by a certification authority):

   ```
   openssl genrsa -out flowforce.key 2048
   ```

   - The private key must be in PEM (Privacy Enhanced Mail) format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
   - In order for the private key to be usable in FlowForce, it must not be password protected, see [Private Key Requirements](#).
   - The private key must be stored securely.

2. Create a Certificate Signing Request (CSR) for the private key generated earlier. You will need the CSR when you purchase your SSL certificate, see the next step. The following OpenSSL command creates a CSR called `myserver.csr` for the key `flowforce.key`:

   ```
   openssl req -new -nodes -key flowforce.key -out myserver.csr
   ```

   When prompted, enter information about your organization, for example:

   ```
   Country Name (2 letter code) [AU]: AT
   State or Province Name (full name) [Some-State]: .
   Locality Name (eg, city) []: Vienna
   Organization Name (eg, company) [Internet Widgits Pty Ltd]: MyCompany Ltd
   Organizational Unit Name (eg, section) []: IT
   Common Name (eg, YOUR name) []: server.my.domain.com
   Email Address []: test@example.org
   ```

   - For the field **Common Name**, make sure to enter the FQDN (fully qualified domain name) of the host machine where FlowForce Server runs.
   - Leave the challenge password field empty when prompted.
3. Order the certificate from a certificate authority. During the ordering process, you will need to supply the CSR. To do this, open myserver.csr in a text editor such as Notepad, copy its contents to clipboard, and then paste it into the online order form.

4. Once the certificate authority validates your company, they will provide to you the purchased certificate and the so-called "intermediary" certificates. Copy-paste the content of all the intermediary certificates into one file, as shown in Preparing Intermediary Certificates.

Summary
If you followed the steps above, you must have by now the following certificates and keys:

- **flowforce.key** - This private key accompanies the certificate used by FlowForce.
- **certificate.crt** (the file extension may vary) - This is your purchased certificate that encrypts the connection between a browser and FlowForce Web Server, or the connection between a client application that connects to a Web service exposed by FlowForce Server.
- **intermediate.pem** - This file includes all the intermediate certificates that you received from the certificate authority.

You can now enable SSL for FlowForce Server, FlowForce Web Server, and for the HTTP connection between them, as shown below:

- [Enabling SSL for FlowForce Web Server](#)
- [Enabling SSL for FlowForce Server](#)
- [Enabling SSL between FlowForce Web Server and FlowForce Server](#)

3.4.1.1 Preparing Intermediary Certificates
When you sign a certificate with a Certificate Authority, you will receive intermediary certificates that form the chain of trust between your server and the certificate authority. To use intermediary certificates in FlowForce Server, you must combine all of them into a single file (the so-called "Certificate Chain File"), as shown below:

1. Using a text editor such as Notepad, create a new text file (let's call it intermediary.pem, you can also choose another file name and extension).
2. Open each intermediary certificate in a text editor, and copy-paste the content into the intermediary.pem file. Importantly, the certificate text must be copied in reverse order (that is, the secondary intermediary certificate goes first, the primary goes second), for example:

```
--BEGIN CERTIFICATE--
... (secondary intermediary certificate) ...
--END CERTIFICATE--
--BEGIN CERTIFICATE--
... (primary intermediary certificate) ...
--END CERTIFICATE
```

3. Save the intermediary.pem file. You will need to refer to it from FlowForce setup page later.
3.4.2 Creating Self-Signed SSL Certificates

This demo shows you how to create self-signed SSL certificates for FlowForce Server running on a private network. Note that this demo is intentionally simplified and not suitable for use in production. Your organization will likely have specific security policies concerning SSL certificates and could use SSL tools other than the ones described below. For information about obtaining SSL certificates signed by a trusted certificate authority, see Signing SSL Certificates with a Certificate Authority.

Prerequisites
This example makes use of the OpenSSL toolkit to generate self-signed certificates. Note that OpenSSL is an open source library, and may need to be compiled before you can use it at the command line. The compilation and installation instructions for OpenSSL vary for each operating system and are outside of the scope of this documentation. On a Linux and Mac machine, it is likely that OpenSSL is already available; otherwise, you can install it or update it from the command line. You can quickly check if OpenSSL present by typing the command below (it displays the current OpenSSL version):

```bash
openssl version
```

On Windows, you can either compile binaries from the official OpenSSL source code, or, alternatively, download a binary distribution that includes OpenSSL. See also https://www.openssl.org/community/binaries.html.

Create the root certificate

1. Create a directory that will store all certificates used in this demo (for example, "C:\secure"). This will be the working directory for all subsequent OpenSSL commands (that is, any file paths are relative to it). Therefore, change to this directory from the command line:

   ```bash
   cd C:\secure
   ```

2. For this demo, we will be issuing certificates with OpenSSL extensions. To make this possible, find the `openssl.cnf` file of your OpenSSL distribution and copy it to the working directory created in the previous step.

3. Create the root private key. Be aware that the root private key is the most sensible piece of your public key infrastructure, so it must always be generated and stored in a secure environment (in this demo, it is stored in "C:\secure").

   ```bash
   openssl genrsa -aes256 -out root.key 2048
   ```

   When prompted, type a password to protect the root key. You will subsequently need this password to sign certificate requests.

4. Create the root certificate. The command below generates a self-signed certificate for the private key created above, with a validity of 3650 days. Notice that the `-config` parameter points to the `openssl.cnf` file in the same directory. The `-extensions` parameter refers to the "v3_ca" extension (section) defined in `openssl.cnf`.

   ```bash
   openssl req -new -key root.key -out root.csr -config openssl.cnf -extensions v3_ca
   ```

   ```bash
   openssl x509 -req -in root.csr -signkey root.key -out root.crt -config openssl.cnf -days 3650
   ```

   The private key and certificate are now complete.

   ```bash
   openssl rsa -in root.key -out root.key.unprotected
   ```

   ```bash
   openssl x509 -in root.crt -out root.crt.unprotected
   ```

   Rename the unprotected files as follows. This will enable signing certificate requests.

   ```bash
   mv root.key root.key.unprotected
   mv root.crt root.crt.unprotected
   ```

   ```bash
   openssl pkcs12 -export -inkey root.key.unprotected -in root.crt.unprotected -out root.pfx
   ```

   ```bash
   openssl pkcs12 -in root.pfx -out root.pkcs12 -nopass
   ```

   The `root.key.unprotected` and `root.crt.unprotected` files are now the protected files.

   ```bash
   openssl pkcs12 -in root.pfx -out root.pfx.unprotected
   openssl pkcs12 -in root.pkcs12 -out root.pkcs12.unprotected
   ```

   Rename the files as follows.

   ```bash
   mv root.pfx root.pfx.unprotected
   mv root.pkcs12 root.pkcs12.unprotected
   ```
Configure the Server

openssl req -config openssl.cnf -extensions v3_ca -x509 -new -nodes -key root.key -sha256 -days 3650 -out root.pem

When prompted, enter information about your organization, for example:

Country Name (2 letter code) [AU]: AT
State or Province Name (full name) [Some-State]: .
Locality Name (eg, city) []: Vienna
Organization Name (eg, company) [Internet Widgits Pty Ltd]: MyCompany Ltd
Organizational Unit Name (eg, section) []: IT
Common Name (eg, YOUR name) []: Demo CA
Email Address []: test@example.org

You can fill in the required fields as applicable to your organization. For the field Common Name, enter the name of your self-signed certificate authority ("Demo CA", in this example).

Create the FlowForce certificate

You can now create the actual certificate to be used for SSL encryption (by FlowForce Server, or FlowForce Web Server, or both). The following OpenSSL command creates the private key:

openssl genrsa -out flowforce.key 2048

Note
- The private key must be in PEM (Privacy Enhanced Mail) format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
- In order for the private key to be usable in FlowForce, it must not be password protected, see Private Key Requirements.
- The private key must be stored securely.

Next, open the working openssl.cnf file and add the following section to it:

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
supportKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName=DNS:server.my.domain.com

Make sure to change the subjectAltName ("Subject Alternative Name") so that it corresponds to the FQDN (fully qualified domain name) of the machine where FlowForce Server runs. In this example, it is set to "server.my.domain.com". Specifying a subject alternative name is required by Google Chrome 58 or later; otherwise, your self-signed certificate will generate a NET::ERR_CERT_COMMON_NAME_INVALID error (see https://support.google.com/chrome/a/answer/7391219?hl=en).
Next, create a Certificate Signing Request (CSR), as shown below. Notice that the -config parameter points to the `openssl.cnf` file edited previously. The -extension parameter refers to the "server_cert" extension defined in `openssl.cnf`.

```
openssl req -config openssl.cnf -extensions server_cert -new -nodes -key flowforce.key -out flowforce.csr
```

When prompted, enter information about your organization, for example:

```
Country Name (2 letter code) [AU]: AT
State or Province Name (full name) [Some-State]: .
Locality Name (eg, city) []: Vienna
Organization Name (eg, company) [Internet Widgits Pty Ltd]: MyCompany Ltd
Organizational Unit Name (eg, section) []: IT
Common Name (eg, YOUR name) []: server.my.domain.com
Email Address []: test@example.org
```

**Note**
- For the field *Common Name*, make sure to enter the FQDN (fully qualified domain name) of the host machine where FlowForce Server runs.
- Leave the challenge password field empty when prompted.

For this demo, we will sign the FlowForce certificate directly with the root certificate. Note that, in a production environment, the root certificate does not normally sign server certificates directly; instead, intermediary certificates are used. The command below signs the `flowforce.csr` certificate request against the root certificate created previously and creates a `flowforce.crt` file (which is the server certificate required in FlowForce Server):

```
openssl x509 -extfile openssl.cnf -extensions server_cert -req -in flowforce.csr -CA root.pem -CAkey root.key -CAcreateserial -out flowforce.crt -days 365 -sha256
```

**Summary**

If you followed the steps above, you must have by now the following certificates and keys:

- **root.key** - This is your certificate authority’s (CA) private key. Store this file in a secure place; if this key becomes compromised, then anyone can generate browser-trusted certificates on your behalf.
- **root.pem** - This is the public certificate of your certificate authority. You will need to install (import) this certificate into the trusted certificates store of each machine (or browser) that needs to access FlowForce securely, see Importing Root Certificates.
- **flowforce.key** - This private key accompanies your self-signed certificate used by FlowForce (see next item).
- **flowforce.crt** - This is a self-signed certificate to be used by FlowForce Server, FlowForce Web Server, or both.

You can now enable SSL for FlowForce Server, FlowForce Web Server, and for the HTTP connection between them, as shown below:

- **Enabling SSL for FlowForce Web Server**
3.4.2.1 Importing Root Certificates

When you create your own certificate authority (CA), the root certificate is self-signed; therefore, no browser will trust it by default. In other words, any browser connecting to FlowForce Server will still display a warning like “This site is not trusted”. In order for an HTTP client (such as a browser) to trust your self-signed certificate, the certificate must be imported as follows:

- Into the operating system's trusted certificates store, if the browser uses the latter. On Windows, for example, Google Chrome and Microsoft Edge use the operating system's certificate store while Mozilla Firefox uses its own store. On Linux, both Google Chrome and Mozilla Firefox use their own certificate store (see next item). On Mac, Safari uses the operating system's certificate store (Keychain Access).
- Into the trusted certificates store of the browser itself.

**Note**
- This step must be performed for each client machine (or browser, if applicable) that will access FlowForce Server.
- When you enable SSL encryption between FlowForce Web Server and FlowForce Server, it is not sufficient to import the certificate into the browser. Your self-signed root CA certificate must be trusted by the operating system.

**Linux**

On Linux, you can import a trusted certificate into the system's certificate store as shown below.

Perform the following steps only if you are sure of the authenticity of the certificate you want to trust.

On Debian and Ubuntu, follow the steps below:

1. Copy the certificate file of the Web server to the following directory.

   ```bash
   sudo cp /home/downloads/server_cert.crt /usr/local/share/ca-certificates/
   ```

2. Update the certificate store as follows:

   ```bash
   sudo update-ca-certificates
   ```

On CentOS, follow the steps below:

1. Install the `ca-certificates` package:

   ```bash
   yum install ca-certificates
   ```

2. Enable the dynamic certificate authority configuration feature:
3. Copy the server certificate to the following directory:
   ```bash
cp server_cert.crt /etc/pki/ca-trust/source/anchors/
   ```

4. Use the command:
   ```bash
   update-ca-trust extract
   ```

For cases where you need to access the server only through the browser, it is sufficient to import the certificate into the browser certificate store. The exact instructions will vary for each browser. For example, in Firefox 59.0.2, you can do this as follows:

1. Under Options | Privacy & Security, click View Certificates.
2. On Authorities tab, click Import and browse for the root certificate file created previously.
3. When prompted, select Trust this CA to identify websites.

   ![Downloading Certificate](image)

   You have been asked to trust a new Certificate Authority (CA).

   Do you want to trust “ODIN CA” for the following purposes?
   - Trust this CA to identify websites.
   - Trust this CA to identify email users.
   - Trust this CA to identify software developers.

   Before trusting this CA for any purpose, you should examine its certificate and its policy and procedures (if available).

   ![View Certificate](image)

 **Mac**

On macOS, you can import a trusted certificate into Keychain Access as follows.

1. Run Keychain Access.
2. Click System, and then click Certificates.
3. On the File menu, click Import Items.
4. Browse for the trusted certificate, and click Open.
5. Enter the Keychain Access password when prompted, and then click Modify Keychain.
6. Double-click the certificate, expand the Trust section, and select Always Trust.
Configure the Server

Setting up SSL Encryption

Configure the Server

Windows

On Windows, you can import a trusted certificate into the system certificates store as follows:

1. Open the Windows certificate store for the computer account, see Accessing Windows Certificate Store.

2. Under “Trusted Root Certification Authorities”, right-click Certificates, and select All Tasks | Import, and follow the certificate import wizard.
3.4.2.2 Accessing Windows Certificate Store

On Windows, you can manage certificates from the Microsoft Management Console (MMC) snap-in, either for your user account, or for the computer account.

**To open the Certificates snap-in (for the current Windows user):**

- Run `certmgr.msc` at the command line.

**To open the Certificates snap-in (for the computer account):**

1. Run `mmc` at the command line.
2. On the **File** menu, click **Add/Remove Snap-in**.
3. Click **Certificates**, and then click **Add**.

4. Click **Computer account**, and click **Next**.
5. Click **Local computer**, and then click **Finish**.
3.4.3 Private Key Requirements

Because FlowForce Server runs unattended, enabling SSL requires that the certificate's private key be unencrypted. In other words, it must not be protected with a password; otherwise, it cannot be used by FlowForce Server. For this reason, the file that stores the private key must have restricted access and be accessible only to entitled personnel in your organization.

To identify whether the private key is password-protected or unencrypted, open the private key file using a text editor or the command line. An encrypted private key begins with the following lines:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,DFC3FAD546517ED6336CFF72AA23F6C7
```

One of the ways to decrypt the private key is by using the OpenSSL toolkit (https://www.openssl.org/). Specifically, you can run the following command to decrypt a private key:

```
openssl rsa -in enc.key -out dec.key
```

Where `enc.key` is the encrypted key and `dec.key` is the unencrypted key to be output by the command. For example, assuming that `ffenc.key` is your encrypted key, you can obtain the unencrypted key by running the following command in the directory where the private key is stored:

```
openssl rsa -in ffenc.key -out ffdec.key
```

After running the command, the `ffdec.key` file no longer states ENCRYPTED, for example:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAzCCedru/oKzaSiwh6avtf9eMPix99RKpd07fWtwstkug1Adi
--
--
--
-----END RSA PRIVATE KEY-----
```

3.4.4 Enabling SSL for FlowForce Web Server

The instructions below show you how to enable SSL for the "FlowForce Web Server" service, that is, the service which drives the Web administration interface of FlowForce.

Prerequisites:

- You need a private key and its corresponding certificate signed by a certificate authority trusted by your browser (such as DigiCert, Comodo, and so on). You also need all the intermediary certificates provided by the certificate authority. For information about obtaining these, see Signing SSL Certificates with a Certificate Authority or Creating Self-Signed SSL Certificates.
- Both the certificate file and the private key must be in PEM (Privacy Enhanced Mail) format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
Configure the Server

Setting up SSL Encryption

- The certificate must be issued for the domain name on which FlowForce Server is running.
- The private key of the certificate must not be encrypted with a password, see Private Key Requirements.
- If you created self-signed certificates, each client browser must be configured to trust your self-signed certificate authority, see Importing Root Certificates.

Once the prerequisites are met, you can secure the connection between a browser and FlowForce Web Server as follows:

1. Open the FlowForce Server setup page.
2. Find the settings grouped under "FlowForce Web Server" and do the following:
   a. Select the Enabled check box under "SSL Encrypted Connection".
   b. Next to "Bind address", select All interfaces (0.0.0.0) (assuming that FlowForce Web Server should be accessible from the outside world, not just locally from the current machine).
   c. Enter the host (domain) name and port where FlowForce Web Server should listen for SSL encrypted connections, in the Host name and Port fields, respectively.
      The domain name entered in the Host name field must correspond to the SSL certificate's Common Name. The port must not be in use.
      Depending on the case, you can also select "other" and enter the IP address where FlowForce Server listens for SSL encrypted connections. If you entered an IP address in the "other" field without entering a host name, this IP address must correspond to the SSL certificate's Common Name.
   d. Enter the path to the certificate and private key file in their respective text boxes.
      Note
      - The certificate must be in PEM (Privacy Enhanced Mail) format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
      - The private key must be in PEM (Privacy Enhanced Mail) format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
      - In order for the private key to be usable in FlowForce, it must not be password protected, see Private Key Requirements.
      - The private key must be stored securely.
   e. Enter the path to the certificate chain file in the Certificate Chain File field. If there is no intermediary certificate, then you can leave this field empty. If there are several intermediary certificates available, then you must combine all of them into a so-called "Chain File" which contains all the intermediary certificates, as described in Preparing Intermediary Certificates.
   f. Optionally, clear the Enabled check box under "Unencrypted Connection". Note that this will make FlowForce Web Server unavailable through plain HTTP, so you should take this step only after the SSL encrypted connection works. Instead of disabling the HTTP connection completely, you may want to restrict it to local connections only, as shown in the image below.
3. Click **Apply settings and restart FlowForce services**.

After you select the SSL **Enabled** check box and click **Apply settings and restart FlowForce services**, the browser will be redirected to the "https" (not the "http") URL.

Note the following:

- The browser (or connecting client) will still display warnings if the **Common Name** (CN) of the SSL certificate does not correspond to the domain name or IP address where FlowForce Server runs.
- If you are using self-signed certificates, the browser (or connecting client) will still display warnings if you did not add your CA root certificate to the operating system's certificate store, or to the browser's certificate store (see Importing Root Certificates).

### 3.4.5 Enabling SSL for FlowForce Server

This topic deals with enabling SSL for the "FlowForce Server" service, that is, the service responsible for exposing Web services created from FlowForce jobs to HTTP(S) clients. If you are looking to enable SSL between a browser and the FlowForce Web administration interface, see Enabling SSL for FlowForce Web Server.

Prerequisites:

- You need a private key and its corresponding certificate signed by a certificate authority trusted by your browser (such as DigiCert, Comodo, and so on). You also need all the intermediary certificates provided by the certificate authority. For information about obtaining these, see Signing SSL Certificates with a Certificate Authority or Creating Self-Signed SSL Certificates.
- Both the certificate file and the private key must be in PEM (Privacy Enhanced Mail) format. The file
extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.

- The certificate must be issued for the domain name on which FlowForce Server is running.
- The private key of the certificate must not be encrypted with a password, see Private Key Requirements.
- If you created self-signed certificates, each client browser must be configured to trust your self-signed certificate authority, see Importing Root Certificates.

Once the prerequisites are met, you can secure the connection between a client machine and FlowForce Server as follows:

1. Open the FlowForce Server setup page.
2. Find the settings grouped under "FlowForce Server" and do the following:
   a. Select the Enabled check box under "SSL Encrypted Connection".
   b. Next to "Bind address", select All interfaces (0.0.0.0) (assuming that FlowForce Server should be accessible from the outside world, not just locally from the current machine).
   c. Enter the host (domain) name and port where FlowForce Web Server should listen for SSL encrypted connections, in the Host name and Port fields, respectively.

   The domain name entered in the Host name field must correspond to the SSL certificate's Common Name. The port must not be in use.

Depending on the case, you can also select "other" and enter the IP address where FlowForce Server listens for SSL encrypted connections. If you enter an IP address in the "other" field without entering a host name, this IP address must correspond to the SSL certificate's Common Name.

d. Enter the path to the certificate and private key file in their respective text boxes.

Note
- The certificate must be in PEM (Privacy Enhanced Mail) format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
- The private key must be in PEM (Privacy Enhanced Mail) format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
- In order for the private key to be usable in FlowForce, it must not be password protected, see Private Key Requirements.
- The private key must be stored securely.

e. Enter the path to the certificate chain file in the Certificate Chain File field. If there is no intermediary certificate, then you can leave this field empty. If there are several intermediary certificates available, then you must combine all of them into a so-called "Chain File" which contains all the intermediary certificates, as described in Preparing Intermediary Certificates.

f. Optionally, clear the Enabled check box under "Unencrypted Connection". Note that this will make FlowForce Server unavailable through plain HTTP.
3. Click **Apply settings and restart FlowForce services**.

Note the following:

- The browser (or connecting client) will still display warnings if the **Common Name** (CN) of the SSL certificate does not correspond to the domain name or IP address where FlowForce Server runs.
- If you are using self-signed certificates, the browser (or connecting client) will still display warnings if you did not add your CA root certificate to the operating system's certificate store, or to the browser's certificate store (see Importing Root Certificates).

### 3.4.6 Enabling SSL between FlowForce Web Server and FlowForce Server

The communication between FlowForce Web Server (FFW) and FlowForce Server (FFS) depends on how you have configured their SSL options, as described previously, namely:

- If you have configured FFS to accept unencrypted connections and disabled SSL, then communication between the two is unencrypted (by default, via port 4646).
- If you have configured FFS for SSL and disabled unencrypted connections, then communication between FFW and FFS is encrypted (by default, via port 4647).

The details of how FFW communicates with FFS are displayed at the top of the Login page, for example:
In the example above, FFW connects to FFS at local address 127.0.0.1, through an encrypted connection on port 4647 (the connection is encrypted assuming that you have enabled SSL for FFS on this port specifically).

You can also change the connection details between FFW and FFS manually, by editing their respective .ini files from the data subdirectory of the FlowForce Server Application Data directory.

Do the following in the flowforce.ini file:

1. In the [ListenSSL] section, enter values for the following parameters:

   ```ini
   [ListenSSL]
   active=1
   ssl=1
   host=0.0.0.0
   port=4647
   hostname=server.my.domain.com
   ```

   - The ssl and active parameters must be set to 1 (enabled).
   - The host must be 0.0.0.0 (all interfaces)
   - The hostname must match the Common Name of the SSL certificate used by FlowForce Server.
   - The port must be other than the default 4646 port used for unencrypted connections. For example, you can set it to 4647, if this port is not already in use.

2. In the [SSL] section, enter the path to the certificate and private key available for FlowForce Server. This is the same certificate and private key pair mentioned in Enabling SSL for FlowForce Server. For example:

   ```ini
   [SSL]
   certificate=C:\secure\flowforce.crt
   private_key=C:\secure\flowforce.key
   certificate_chain=
   ```

Do the following in the flowforceweb.ini file:

1. If it does not exist already, add a section called [FlowForce], and type values for the following parameters:

   ```ini
   [FlowForce]
   host=127.0.0.1
   ```
Configure the Server

| port=4647  
| ssl=1      
| certificate=C:\secure\flowforce.crt |

- The `ssl` parameter must be enabled (set to 1).
- The `host` in this case is 127.0.0.1 since the communication between FFS and FFW is local.
- The `port` must point to the port where FFS accepts encrypted connections (4647, by default).
- The `certificate` defines the local path to the FFS certificate file (or the path to the common certificate of FFS and FFW, if both are using the same).

**Note:** After you finished editing the .ini files, restart both the FlowForce Server and the FlowForce Web Server services. For more information, see:

- Starting and Stopping Services (Linux)
- Starting and Stopping Services (macOS)
- Starting and Stopping Services (Windows)
3.5 Setting the Default Time Zone

Whenever you create jobs that use time-based triggers, you must specify the applicable time zone. For convenience, you can configure globally what time zone should be selected by default in the job configuration page.

To set the default time zone:

1. Log on to the FlowForce Web administration interface.
2. Click Administration.
3. Click Settings.
4. Under Input format, select the default time zone.
5. Click Save.
3.6 Setting Mail Parameters

If you are creating jobs that send emails, you need to configure the SMTP address and port of the mail server, as well as the SMTP credentials.

FlowForce will first attempt to establish a connection encrypted over TLS or SSL. If the encrypted connection fails, FlowForce attempts to start communication without encryption, and then might elevate connection to encrypted if the SMTP server would explicitly require it. Otherwise, the SMTP connection remains in plain text.

To change the mail settings:

1. Log on to the FlowForce Web administration interface.
2. Click Administration.
3. Click Settings.
4. Under SMTP Server and SMTP port, enter the name and port of the mail server, respectively. Standard SMTP servers accept connections on port 25. SMTP servers that require connection to be encrypted over SSL/TLS protocol accept connections on other ports, typically 465 or 587.
5. If your SMTP server requires authentication, click the plus icon next to User authentication and enter the username and password.
6. Optionally, enter a RFC2822–compliant mailbox address value in the Default Sender field. The value entered here is used as the default From parameter of the /system/mail/send and /system/mail/send-mime functions.
3.7 Directory Service Settings

If your organization uses Microsoft Active Directory or an LDAP-compliant directory service provider such as Apache Active Directory, OpenLDAP Server, Oracle Unified Directory, and others, you can integrate it with FlowForce Server. From the FlowForce Server perspective, integration with a Directory Service provider means the following:

- Users can log on to FlowForce Server with their domain user name and password.
- Administrators can either allow existing domain users to log on to FlowForce Server with their domain credentials (that is, an implicit user import takes place), or they can explicitly import domain users and groups into FlowForce Server (see Importing Domain Users and Roles). In either case, the imported accounts are visible in the user administration pages of FlowForce Server. This enables administrators to add or revoke privileges and permissions to groups or user accounts, in the same way as for the built-in FlowForce Server accounts (see How Privileges Work and How Permissions Work). Administrators can also assign FlowForce Server roles to groups or user accounts (see Assigning Roles to Users).
- Administrators cannot rename or change the password of domain users imported into FlowForce Server.
- Administrators cannot rename or change the membership of domain groups imported into FlowForce Server.
- Administrators can delete imported domain accounts from FlowForce Server. This does not remove the accounts from the domain and does not change in any way their associated domain privileges.
- If the imported domain accounts have FlowForce Server privileges and permissions assigned to them, they are displayed in privilege reports (see Viewing Privilege Reports).

To change the Directory Service settings:

1. **Log on** to the FlowForce Web administration interface.
2. Go to the Administration menu and click Settings.

![Directory Service Configuration](image-url)
Configure the Server
Directory Service Settings

The available settings are described below.

Enable
Select this check box to enable users to log on to FlowForce Server with their domain user name and password. If you select this check box, you must select either the Active Directory or the Lightweight Directory Access Protocol (LDAP) option, as further described below.

If you select the Lightweight Directory Access Protocol (LDAP) option, make sure that connection details (such as username, password, and so on) are correct. When you click Save, FlowForce attempts to communicate with the specified LDAP server and shows an error if the connection details are not valid. Note that FlowForce Server must be able to connect to the LDAP server successfully before you can save the LDAP settings.

If you select the Active Directory option, the machine where FlowForce Server runs must be part of a domain controlled by Active Directory.

After you have enabled directory service authentication, an additional drop-down list becomes visible in the FlowForce Server login page, called Login. The Login drop-down list enables users to select the authentication option and contains the following items:

- Directly. This is the default FlowForce Server authentication option. To log in, users must supply their FlowForce username and password.
- [A specific domain], depending on the configured LDAP server. To log in, users must supply their domain username and password—these are managed by the LDAP server.

See also Logging on to FlowForce Server.

Connect using
Select Active Directory to enable direct integration with Active Directory. This is applicable if FlowForce Server runs on Windows and the machine is part of a domain controlled by Active Directory.

Select Lightweight Directory Access Protocol (LDAP) to enable integration with an LDAP-compliant Directory Service. Fill in the details as follows:

- Host — Enter the host name, domain name, or IP address of the LDAP server. To add a port number, append a colon character, followed by the port number. For example, somehost:10389
- User — Enter a user name which has administrative rights to query the directory service. The user name can either be in the form of a "Distinguished-Name" (for example, cn=name,dc=domain,dc=com) or a "User-Principal-Name" (for example, user@some.domain.com). Note: The "User-Principal-Name" format applies for Active Directory only; for other LDAP servers, use the "Distinguished-Name" format.
- Password — The user's password. Note: If you mistype the password several times, the LDAP server may lock the account. In that case, make sure that the account is not locked out before proceeding.
- Use SSL — Select this check box only if the LDAP server was configured to accept SSL-encrypted connections from clients. If you select this option, change the port number to the one used by the LDAP server for secure connections (typically, port 636). If your organization already uses the same trusted root certificate on both machines, there are typically no additional configuration instructions. Otherwise, the root (CA) certificate of the LDAP server must be installed on the machine where FlowForce Server runs, as follows:
Configure the Server Directory Service Settings

a. On the machine where LDAP server is, export the root certificate from the trusted certificate store. Use the tools specific to your operating system for that purpose (for example, the Certificates Snap-In on Windows).

b. On the machine where FlowForce Server is, import the certificate into the trusted certificate store, as described in Importing Root Certificates.

In some cases, LDAP servers can have arbitrary schemas that do not fit into a particular standard. If FlowForce Server cannot detect the schema of your LDAP provider, an error similar to “Directory Service detected an invalid LDAP schema” is displayed. In this case, copy the directoryservice.cfg file to the same directory as the FlowForce Server executable. When this file is present, FlowForce Server will not attempt to detect the schema of the LDAP provider automatically.

Allow any domain users to log in

Select this check box if a user’s domain account should be imported into the FlowForce user database first time when users log on to FlowForce with their domain credentials. If this option is disabled, domain users can log on to FlowForce Server only if their account has already been imported into FlowForce Server by an administrator. See Importing Domain Users and Roles.

Default login domain

This option is visible after the Enable check box is selected and the settings have been saved.

The drop-down list displays all domains that this machine is member of. The same list of domains will be visible to users in the FlowForce login page, if Directory Service authentication is enabled (see the first option above).

Select the Set domain login as default check box if the domain should be selected as the default choice in the Login drop-down list of the FlowForce Server authentication page.

If you clear the Set domain login as default check box, the built-in FlowForce Server authentication (“Directly”) is the default choice.
3.8 Logging Settings

FlowForce Server provides a logging mechanism to register all kinds of events and the time when they occurred (such as job outcome events, configuration change events, errors, and so on). You can view all the log events from a dedicated page, see Viewing the Job Log. Note that the log events can significantly increase the size of the FlowForce Server internal database over time. For this reason, the log must be periodically archived or cleaned up using the archive-log or truncate-log/system/maintenance functions. There are also other settings available that help you keep the size of the log within reasonable limits, as further described on this page.

The logging that takes place in FlowForce Server can be of two types:

1. Default system logging that does not require manual intervention of any kind. This kind of logging is taken care of by the system and does register all events, but keeps the size of each log record up to a certain limit, for better system stability and performance. If the system logging does not provide enough level of detail, or if you find out that certain log entries (such as parameter values in steps) are truncated because they are too long, you can use explicit logging, as described next.

2. Optional (explicit) logging that you can enforce from the job configuration page. The job configuration page provides a Log button that you can optionally enable next to each parameter which you are interested to track in the log. Doing so will log the full value of that parameter when the job runs. In addition, you can embed any FlowForce expression inside the log expression function in order to request that that expression be logged explicitly. Again, this will log the expression in full and its value will not be truncated. FlowForce Server does not limit the size of entries logged as a result of explicit logging.

Logging limits

If you do not want to use explicit logging for whatever reason, you can alternatively change the default size of log entries maintained by the system.

Changing the default log size to a higher value may impact system stability and performance, so exercise this option carefully. The recommended approach is to use explicit logging, as mentioned above.

To view or change the default size of log entries:

1. Log on to the FlowForce Web administration interface.
2. Go to Administration | Settings and observe the parameters grouped under “Logging limits”.

<table>
<thead>
<tr>
<th>Logging limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>String type limits:</td>
</tr>
<tr>
<td>Default string value logging limit: 50 characters</td>
</tr>
<tr>
<td>List type limits:</td>
</tr>
<tr>
<td>Default list value logging limit: 10 items</td>
</tr>
<tr>
<td>Recurse into sublists: 2 levels</td>
</tr>
</tbody>
</table>
Configure the Server Logging Settings

Notice that there are two kinds of logged entries: string types and list types. Consequently, there are two parameters to control the size of each type.

<table>
<thead>
<tr>
<th>Default string value logging limit</th>
<th>Specifies the default length of log entries that are of type &quot;string&quot;. If a log entry exceeds this value then long arbitrary values such as file paths will be truncated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default list value logging limit</td>
<td>Same as above, applies to log entries that are of type &quot;list&quot;.</td>
</tr>
<tr>
<td>Recurse into sublists</td>
<td>This setting affects jobs which operate on lists that contain other lists as children. Set this value to instruct FlowForce to look N levels deep for logging purposes.</td>
</tr>
</tbody>
</table>

Instance logging

The settings in the "Instance logging" section specifically affect the level of information reported in the Instance log page.

Logged messages can have severity levels, in this order (from lowest to highest): information, warning, error. The "Instance logging" parameters make it possible to skip logging of certain messages according to their severity. You can also configure the amount of tracing information that should be stored by FlowForce Server, or completely disable retention of logs. The image below illustrates the default settings:

![Instance logging](image)

Clearing the Retain log check box has the effect that no information is reported at all in the Instance log page.

The Messages severity option specifies what messages should be retained:

<table>
<thead>
<tr>
<th>None</th>
<th>No messages are kept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Keep errors and critical messages</td>
</tr>
<tr>
<td>Warning</td>
<td>Keep errors, critical messages, and warnings</td>
</tr>
<tr>
<td>Information</td>
<td>Keep errors, critical messages, warnings, and information messages</td>
</tr>
<tr>
<td>All</td>
<td>This is the most verbose option. All possible messages are kept, regardless of their severity.</td>
</tr>
</tbody>
</table>
The **Execution trace** parameter specifies the amount of tracing detail that should be stored:

<table>
<thead>
<tr>
<th>None</th>
<th>No tracing information is kept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams</td>
<td>Keep streams but exclude traces</td>
</tr>
<tr>
<td>Trace</td>
<td>Keep traces but exclude streams</td>
</tr>
<tr>
<td>Full</td>
<td>Keep every possible level of tracing information.</td>
</tr>
</tbody>
</table>

**Rules**

The "Instance logging" settings described above constitute a "rule". You can create custom rules, in addition to the default one, by clicking the button. This makes it possible to apply rules conditionally, based on the outcome of the job, which can be one of the following:

<table>
<thead>
<tr>
<th>Successful execution</th>
<th>The job is considered successful.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failed execution</td>
<td>The job execution has failed.</td>
</tr>
<tr>
<td>Stopped by user</td>
<td>The job was stopped by user action, see <em><a href="#">Stopping Jobs</a></em>.</td>
</tr>
<tr>
<td>Interrupted</td>
<td>The service was stopped before the job could finish, or FlowForce Server crashed.</td>
</tr>
</tbody>
</table>

The rules defined on this page are evaluated from top to bottom. If the job outcome matches *any* of the outcomes listed above, the rule is matched. The first matching rule wins.

For example, the configuration illustrated below retains the full message log if the job execution was not successful. In other words, the first rule will be triggered if the outcome is "Failed execution" or "Stopped by user" or "Interrupted". On successful execution, the "Default" rule will be triggered instead, and, even though the log messages will be kept, no tracing information will be available.
Note that you can add all the custom rules only before the default rule, not after it. To change the order of rules, use the Up and Down buttons. These buttons are enabled only when there are three or more rules.

If you define custom rules, it is advisable to use the default rule as a "catch all" filter, in case none of the rules before it has matched.

Logging rules at object level
You can create logging rules not only globally at application level, but also for specific FlowForce Server jobs. Note that, if you create a rule on a job that has sub-jobs, then the rule will apply to all the sub-jobs as well.

To set logging rules for a job:

1. Open the job configuration page.
2. Click the Logging Settings button in the job configuration page.
3. Click New Instance Logging rule.

To set logging rules for multiple jobs:

1. Click Configuration and open a container.
2. Select one or more jobs (or the entire container), and then click Logging settings for selected jobs. A dialog box appears where you can refine the selection if necessary:
3. Click Change Logging Settings.
4. Click New Instance Logging rule.

All the logging configuration settings work in the same way as described above in the “Rules” section.
If you have defined logging rules both at object level and at application level, then the priority is established as follows:

- The logging rules defined at object level are checked first.
- If there is a match found at this level, the rule is applied and the rules at application level are no longer checked.
- If there is no match found at this level, the rules at application level are checked.
3.9 Starting and Stopping Services (Linux)

The FlowForce Server solution consists of two services:

1. flowforcewebserver
2. flowforceserver

Run the commands below to start or stop the flowforcewebserver service. If you need to start or stop the flowforceserver service, replace flowforcewebserver with flowforceserver in the commands below.

To start the FlowForce Web Server service:

```
sudo systemctl start flowforcewebserver
```

To stop the FlowForce Web Server service:

```
sudo systemctl stop flowforcewebserver
```

To check if a service is running, run the following command (replace servicename with either flowforcewebserver or flowforceserver).

```
sudo service servicename status
```
3.10 Starting and Stopping Services (macOS)

To start the FlowForce Server service:

```bash
sudo launchctl load /Library/LaunchDaemons/com.altova.FlowForceServer.plist
```

To start the FlowForce Web Server service:

```bash
sudo launchctl load /Library/LaunchDaemons/com.altova.FlowForceWebServer.plist
```

To stop the FlowForce Server service:

```bash
sudo launchctl unload /Library/LaunchDaemons/com.altova.FlowForceServer.plist
```

To stop the FlowForce Web Server service:

```bash
sudo launchctl unload /Library/LaunchDaemons/com.altova.FlowForceWebServer.plist
```
3.11 Starting and Stopping Services (Windows)

By default, the FlowForce Server services are automatically started when Windows starts. Follow the instructions below if you need to manage services manually.

To start the FlowForce Server service:

- Click the ServiceController icon ( ) in the system notification area, and then select Altova FlowForce Server > Start service.

To start the FlowForce Web Server service:

- Click the ServiceController icon ( ) in the system notification area, and then select Altova FlowForce Web > Start service.

To stop the FlowForce Server service:

- Click the ServiceController icon ( ) in the system notification area, and then select Altova FlowForce Server > Stop service.

To stop the FlowForce Web Server service:

- Click the ServiceController icon ( ) in the system notification area, and then select Altova FlowForce Web > Stop service.

You can also start or stop the FlowForce Server services using the Microsoft Management Console (found under Control Panel > Administrative Tools > Services).
3.12 FlowForce Server Application Data

This topic describes the contents of the FlowForce Server application data directory. This information can be useful during manual data migration to a major FlowForce Server version or if you want to change some of the FlowForce Server configuration settings by editing .ini files.

The application data directory stores data generated by both FlowForce Server and its users, such as jobs, triggers, system functions, server logs, and other files.

The application data directory also contains several .ini style configuration files. Administrators can edit the .ini configuration files with a text editor, as an alternative to changing settings from the Web Administration Interface, the Setup page, or from the Command Line Interface.

The path to the application data directory depends on the operating system and platform and is as follows.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>/var/opt/Altova/FlowForceServer2022</td>
</tr>
<tr>
<td>macOS</td>
<td>/var/Altova/FlowForceServer2022</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\ProgramData\Altova\FlowForceServer2022</td>
</tr>
</tbody>
</table>

The following table lists the main files and folders in the application data directory.

<table>
<thead>
<tr>
<th>File/Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flowforceserver.ini</td>
<td>Stores global configuration settings of FlowForce Server (currently, the language used in server logs and in error messages).</td>
</tr>
<tr>
<td>flowforceweb.ini</td>
<td>Stores global configuration settings of FlowForce Web Server (currently, the language of the Web administration interface). <strong>Note:</strong> Do not confuse this file with the flowforceweb.ini file stored in the data directory (described below).</td>
</tr>
<tr>
<td>data/cache.db</td>
<td>This database file stores data related to the caching feature of FlowForce (see Caching Job Results).</td>
</tr>
<tr>
<td>data/state.db</td>
<td>This database file stores the volatile (that is, not configured) state of FlowForce.</td>
</tr>
<tr>
<td>data/ffweb.log</td>
<td>These files store the log of messages from FlowForce Web Server and FlowForce Server, respectively. This is applicable to Windows and macOS systems. On Debian, Ubuntu and CentOS or later, the log is written to the system log (/var/log/syslog).</td>
</tr>
<tr>
<td>data/flowforce.log</td>
<td>This database file stores the FlowForce Server object system, user data, active jobs, roles, and others.</td>
</tr>
<tr>
<td>data/flowforcelog.db</td>
<td>This database file stores the FlowForce Server logs.</td>
</tr>
<tr>
<td>Directory</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>data/flowforce.ini</td>
<td>This configuration file defines the port and listening interfaces of FlowForce Server. See also <a href="#">Configuration File Reference</a>.</td>
</tr>
<tr>
<td>data/flowforceweb.ini</td>
<td>This configuration file defines the port and listening interfaces of FlowForce Web Server. See also <a href="#">Configuration File Reference</a>.</td>
</tr>
<tr>
<td>data/files</td>
<td>Stores files associated with deployed functions.</td>
</tr>
<tr>
<td>data/logs</td>
<td>Contains captured output from job execution steps.</td>
</tr>
<tr>
<td>data/tmp</td>
<td>Stores temporary files.</td>
</tr>
<tr>
<td>data/tools</td>
<td>When other Altova server products (such as MapForce Server or StyleVision Server) are installed alongside FlowForce Server, this directory stores .tool files which enable FlowForce Server to work with these products. If this directory is empty, you can copy manually any tool files from the installation directory of the respective product to this directory. You may need to edit a .tool file in order to set environment variables that may be required to execute MapForce mappings or StyleVision transformations (see <a href="#">Environment Settings</a>).</td>
</tr>
</tbody>
</table>
3.13  Localizing FlowForce Server

FlowForce Server is delivered with support for the following languages: English, French, German, Spanish, and Japanese. So you do not need to create a localized version of these languages. To set any of these languages as the default language, use FlowForce Server's setdeflang command.

To create a localized version of FlowForce Server:

1. Generate an XML file containing the resource strings by using the exportresourcestrings command.
2. Translate the resource strings into the target language. The resource strings are the contents of the `<string>` elements in the XML file. Do not translate variables in curly brackets, such as `{option}` or `{product}`.
4. After you receive your localized DLL file from Altova Support, save the DLL in the `<INSTALLATION FOLDER>\FlowForceServer2022\bin` folder. Your DLL file will have a name of the form `FlowForceServer2022_lc.dll`. The `lc` part of the name contains the language code. For example, in `FlowForceServer2022_de.dll`, the `de` part is the language code for German (Deutsch).
5. Run the setdeflang command to set your localized DLL file as the FlowForce Server app to use. Use the language code that is part of the DLL name as the argument of the setdeflang command.
3.14 Backup, Data Recovery and Migration

This section explains how to perform a backup in FlowForce Server, restore data, and copy FlowForce Server data from a previous application data directory to the current one.

3.14.1 Backup

This topic explains how to back up data in FlowForce Server. In this case, there are two possible options:

- From the Web administration interface. This type of backup includes only configuration data: jobs, credentials, deployed MapForce mappings or StyleVision transformations, resources. It does not include application settings or users. Any FlowForce Server user can import or export configuration data if their permissions on the respective object allow it.

- Administrative backup of the application data directory. This approach requires access to the FlowForce Server application data directory on the machine where FlowForce Server is installed. The application data directory includes all the data from the previous bullet as well as users and roles, including users and roles imported from a Directory Service such as Active Directory. The application data directory also includes application-level settings, such as email or LDAP server settings, password policies.

Note: This topic does not cover backup and recovery of data external to FlowForce Server, such as files or directories that are input/output to jobs, FlowForce resources or local file-based databases. You will need to back up this data separately. It is recommended to keep all such external data (if possible) in the same directory for easier backup and maintenance.

Useful tips

In case you want to migrate data to a new machine in the future or restore it from a backup, the tips below will help you carry over data more easily:

- It is recommended to configure LicenseServer to have a fallback second server (the so-called “failover” server). For details, see the LicenseServer documentation (https://www.altova.com/documentation).

- It is recommended that all jobs should use standalone (not inline) credentials. If you are using local (inline) credentials in jobs, all such jobs will have to be edited on a new server machine to match the user credentials linked to that operating system. By contrast, if you are using standalone credentials, you will only need to edit the standalone credentials on the new server machine.

- If you are running mapping functions deployed from MapForce, consider referring to file and folder paths and databases using resources instead of absolute references.

- As an alternative to creating and maintaining users and roles directly in FlowForce Server, you might want to use Windows Active Directory or another LDAP Server with support for Directory Services. For details, see Changing the Directory Service Settings.

Partial backup from the Web administration interface

To perform a backup of selected objects, log in to the FlowForce Web administration interface and use the Export functionality. To restore data, use the Import functionality. For details, see Importing and exporting configuration data.
Note: You can import configuration data into a FlowForce Server instance that is of the same or later version than the one from where data was exported. Importing configuration data into an earlier version of FlowForce Server may work but should be avoided.

Backup of all FlowForce application data
The backup of all application data involves creating a copy of the FlowForce Server database (DATADIR) in a safe location from where you can later restore it, if necessary.

In the instructions below, DATADIR refers to the following directory:

- Linux: /var/opt/Altova/FlowForceServer2022/data
- macOS: /var/Altova/FlowForceServer2022/data
- Windows: C:\ProgramData\Altova\FlowForceServer2022\data

To save time and disk space, you will want the DATADIR directory to be as compact as possible. You can achieve this by performing the following optional steps before the actual backup:

1. Archive the old log records by creating a job that runs the built-in archive-log function.
2. Delete old log records by creating a job that runs the built-in truncate-log function.
3. Delete unused files by creating a job that runs the built-in cleanup-files function.
4. Run the FlowForce Server executable with the compactdb.

You can now proceed to the actual back-up steps:

1. Stop both the FlowForce Server and FlowForce Web Server services. See the instructions for Linux, macOS, and Windows.
2. Create a copy of DATADIR to a safe directory (preferably on a different machine or disk). By convention, we will call this copy DATADIR_BACKUP in subsequent steps.

The private.db file inside DATADIR contains sensitive information, such as passwords and private keys. Ensure the backup is stored in a secure location.

3.14.2 Data Recovery and Migration

This topic explains how to restore data in FlowForce Server. It also provides information about data migration, which allows copying FlowForce Server data from a previous application data directory to the current one. If necessary, it also upgrades the FlowForce database to the latest version. The migratedb command, which is used to migrate data, can be invoked to copy application data from one folder to another. Running this command may be useful when migrating FlowForce Server to a new machine or when restoring the application data directory from a backup.

If you only need to upgrade the FlowForce database version to the latest one, it is sufficient to run upgradedb.
Restoring data

If the DATADIR_BACKUP is of the same version and on the same machine as the currently running FlowForce Server, you can easily restore it as follows:

1. If FlowForce Server services are running, stop them. See the instructions for Linux, macOS, and Windows.
2. Rename DATADIR, for example, to temp_data.
3. Copy DATADIR_BACKUP to DATADIR.

You can also restore backups that originate from another machine and perhaps have an older database version. The steps below could be useful, for example, if you want to migrate FlowForce data to a new server, or if a hardware failure has occurred.

Note that you can restore data on a machine that runs the same or a different operating system. In the latter case, note that all the paths used in jobs may not be valid on the new operating system, in which case they will need to be updated manually. Importantly, credentials that are tied to operating system user accounts, that is, credentials where the Allow usage for job execution option is enabled, may no longer be valid on a new machine, in which case they will need to be updated manually.

To restore data to a new FlowForce Server installation or version:

1. Install FlowForce Server and any of the following, if applicable: MapForce Server, StyleVision Server and RaptorXML Server. If you need to install LicenseServer as well, you can select it as part of FlowForce Server installation (Windows only). On other platforms, you will need to install LicenseServer separately.
2. Log on to the LicenseServer Web administration interface and deregister all the products from the old machine. Next, register all the products from the new machine with LicenseServer. This step can also be performed after migration.
3. If FlowForce Server services are running, stop them. See the instructions for Linux, macOS, and Windows.
4. Rename DATADIR, for example, to temp_data.
5. Run the migratedb command by supplying DATADIR as --datadir, and DATADIR_BACKUP as --olddatadir, for example:

   Windows
   
   FlowForceServer migratedb
   --datadir=C:\ProgramData\Altova\FlowForceServer2022\data
   --olddatadir=C:\transfer\backup_data

   CentOS
   
   sudo ./flowforceserver migratedb
   --datadir=/var/opt/Altova/FlowForceServer2022/data
   --olddatadir=/home/chang/backups/data

Data migration on Windows

On Windows, you do not typically need to migrate configuration data manually. When you install a new major version of FlowForce Server, and a previous major version is already installed, the installation wizard prompts you to migrate the configuration data.

Should you need to migrate configuration data manually, follow the instructions below:

1. Ensure that Altova ServiceController is running in the system notification area. Otherwise, start the Altova ServiceController.
3. Delete the FlowForce Server data folder installed by the 2022 installation wizard. The path to the data folder depends on your Windows version (see How FlowForce Server Stores Configuration Data).
4. At the command prompt, run the FlowForce executable with the migratedb command, for example:

```
"C:\Program Files(x86)\Altova\FlowForceServer2022\bin\FlowForceServer.exe" migratedb
--datadir=C:\ProgramData\Altova\FlowForceServer2022\data
--olddatadir=C:\ProgramData\Altova\FlowForceServer2020\data
```

Data migration on Linux

Before migrating data:

1. Uninstall the previous version of FlowForce Server. Note that deinstallation does not remove the application data directory. For more information, see Important Paths. The path to the application data directory depends on the major version of FlowForce Server (for example, /var/opt/FlowForceServer2020).
2. Install FlowForce Server 2022. This creates a new application data directory with the default configuration data (for example, /var/opt/FlowForceServer2022).

To migrate data to FlowForce Server 2022:

1. Stop the FlowForce Web Server service if it is running:

   ```
   sudo systemctl stop flowforcewebserver
   ```
2. Stop the FlowForce Server service if it is running. Use the same command as above but replace flowforcewebserver with flowforceserver.
3. Remove or rename the NEW data directory created during the installation:

   ```
   sudo rm -rf /var/opt/Altova/FlowForceServer2022/data
   ```
4. Migrate the EXISTING data by running the migratedb command available in the command-line interface of FlowForce Server. For example:

   ```
   sudo /opt/Altova/FlowForceServer2022/bin/flowforceserver migratedb
   --olddatadir=/var/opt/Altova/FlowForceServer2020/data
   --datadir=/var/opt/Altova/FlowForceServer2022/data
   ```
5. Start the FlowForce Web Server service:
6. Start the FlowForce Server service. Use the same command as above but replace `flowforcewebserver` with `flowforceserver`.

Data migration on macOS

**Prerequisites:**
- FlowForce Server 2022 must be installed (see [Installation on macOS](#)).
- Perform data migration as a user with administrative (root) privileges.

**To migrate data to FlowForce Server 2022:**

1. Stop the FlowForce Server service:
   
   ```
   sudo launchctl unload /Library/LaunchDaemons/com.altova.FlowForceServer.plist
   ```

2. Stop the FlowForce Web Server service:
   
   ```
   sudo launchctl unload /Library/LaunchDaemons/com.altova.FlowForceWebServer.plist
   ```

3. Remove or rename the data directory that was created during the installation. This will delete any objects in the new version that were created after the installation before performing this manual upgrade.
   
   ```
   sudo rm -rf /var/Altova/FlowForceServer2022/data
   ```

4. Run the `migratedb` command:
   
   ```
   sudo /usr/local/Altova/FlowForceServer2022/bin/FlowForceServer migratedb
   --olddatadir=/var/Altova/FlowForceServer2020/data
   --datadir=/var/Altova/FlowForceServer2022/data
   ```

5. Start the FlowForce Server service:
   
   ```
   sudo launchctl load /Library/LaunchDaemons/com.altova.FlowForceServer.plist
   ```

6. Start the FlowForce Web Server service:
   
   ```
   sudo launchctl load /Library/LaunchDaemons/com.altova.FlowForceWebServer.plist
   ```
4 Manage User Access

This section describes procedures and concepts applicable to user access management in FlowForce Server.
4.1 Users and Roles

This section includes the following topics:

- Create Users
- Create Roles
- Import Domain Users and Roles
- Default Users and Roles
- Rename Users and Roles
- Assign Roles to Users
- Assign Roles to Other Roles
- Reset the Root Password

4.1.1 Create Users

Users are persons who log on to FlowForce Server to configure jobs, deploy MapForce or StyleVision transformations, or manage the FlowForce Server. The actions available to users in FlowForce Server depend on the following:

a) Their assigned permissions or privileges
b) The permissions and privileges assigned to any roles that users are members of.

To add a FlowForce Server user:

1. Click Administration, and then click Users.
2. Click Create User.
3. Fill in the required fields.

<table>
<thead>
<tr>
<th>User name</th>
<th>Enter the name of the user. The following restrictions apply:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• It must not be empty</td>
</tr>
<tr>
<td></td>
<td>• It must not begin with or end with spaces</td>
</tr>
<tr>
<td></td>
<td>• The allowed characters are letters, digits, underscore ( _ ),</td>
</tr>
<tr>
<td></td>
<td>dash ( - ), and full stop ( . )</td>
</tr>
</tbody>
</table>

| Password | Enter the user's password.                                    |
| Re-type password | Re-type the user's password.                                 |
| Change password on next login | If you select this check box, the user will be prompted to change password on next login. |

4. Optionally, grant the required privileges to the user (for the description of available privileges, see Privileges). Note that you can grant privileges to users either directly from this page, or by assigning to them a role which already has some privileges. To simplify user maintenance, it is recommended to use the latter approach (see Adding Roles and Assigning Roles to Users).

5. Click Save.
4.1.2 Create Roles

Roles are named sets of privileges that help enforce security based on the business need. The typical role-based security involves at least two roles: an administrator and a standard user. Each role is defined by the privileges granted to that role. For example, administrators can change their own password and that of other users, whereas standard users can change only their own password. You can assign roles to users and revoke roles from users as necessary.

To add a FlowForce Server role:

1. Click Administration, and then click Roles.
2. Click Create Role.
3. Enter the role name (for example, “Administrator”).
4. Under Privileges, select the privileges that must be assigned to the role (for the description of available privileges, see Privileges).
5. Click Save.

4.1.3 Import Domain Users and Roles

In addition to creating FlowForce Server users, you can import domain user accounts and groups from Windows Active Directory or an LDAP Directory Service provider. When the setting Allow any domain users to log in is enabled in the Directory Service settings, users from configured domains are able to log on to FlowForce Server even if you have not explicitly imported their accounts into the FlowForce Server database. To ensure that domain users log on to FlowForce Server only if their account has been explicitly imported by an administrator, clear the check box Allow any domain users to log in and import the domain users, as shown below.

Note: The local machine accounts are not part of Active Directory. Therefore, they cannot be imported into FlowForce Server.

Import domain users
To import domain user accounts into FlowForce Server, take the following steps:

1. Go to Settings and select the Enable check box under Directory Services and configure your preferred Directory Service provider, as described in Changing the Directory Service Settings.
2. In the Administration menu, click Users.
3. Click Import Domain Users.
4. If applicable, select the domain of choice from the **Context** drop-down list.

5. In the **Search for** text box, start typing the name of the user account you want to import. Partial searches are valid: For example, if you enter a value such as *ad*, the accounts *Administrators*, *Admanager*, and *Admin* are retrieved from the LDAP server or Active Directory and shown in the webpage dialog. In the case of Active Directory, FlowForce Server uses the Ambiguous Name Resolution (ANR) search algorithm that allows you to specify complex search conditions in a single clause. For example, you can retrieve the account of a person named Jim Smith by typing *ji sm*. See the Microsoft documentation for further information about Ambiguous Name Resolution in Active Directory.

6. Select the records that you want to import and click **Import Selected**. Waiting time increases if the domain is not local.

### Import domain roles

To import domain groups into FlowForce Server, take the following steps:

1. Click **Roles** in the **Administration** menu.
2. Click **Import Domain Roles**.
3. Follow the steps 4-6 above.

### Domains and domain trusts

You can see the list of available domains on the login page and in the following sections of the **Administration page**: (i) in the dialog box **Import Domain Users** in the **Users** tab, (ii) in the dialog box **Import Domain Roles** in the **Roles** tab, and (iii) in the **Settings** tab. Currently, only the following domains are visible in FlowForce Server: the domain with the machine on which FlowForce Server is installed and any domains from the same *forest* to which this machine belongs. However, other trusted domains connected via the *external*. 
forest, realm and shortcut trusts are not supported and cannot be seen in the list of available domains in FlowForce Server.

**Note:** To run a job, you can use any user credentials accepted by Windows. In this case, Windows will take care of the external trusts.

## 4.1.4 Default Users and Roles

### Default Users

The following special users are predefined in FlowForce Server.

<table>
<thead>
<tr>
<th>User</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>This user is the initial, top-level FlowForce Server administrator. By default, it has all permissions and privileges available in the system.</td>
</tr>
<tr>
<td>anonymous</td>
<td>This is a special user account for users that do not explicitly log in. Anonymous access to the FlowForce Server Administration Interface is not possible, but you can enable anonymous access for certain services exposed as Web services (see <em>Exposing Jobs as Web Services</em>).</td>
</tr>
</tbody>
</table>

The built-in users cannot be deleted, although it is possible to change their privileges.

**Note:** User root can change any privileges and permissions, including own permissions and privileges. Take extra caution when logged in as root and editing root privileges, since you may unintentionally lose your own access to the system. In the event that this happens, see *Resetting the Root Password*.

### Default Roles

The following special roles are predefined in FlowForce Server.

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authenticated</td>
<td>This role includes all users who are authenticated using an existing user name and password. Every FlowForce Server user except user anonymous is a member of this role. By default, this role has the Set own password privilege.</td>
</tr>
<tr>
<td>all</td>
<td>This role includes all FlowForce Server users, including user anonymous. By default, this role has no privileges.</td>
</tr>
</tbody>
</table>

Since the roles authenticated or all are built-in, you cannot explicitly assign these roles to users or revoke them from users. The membership of the built-in roles is automatically managed by FlowForce Server. Every time when you add a new user, FlowForce Server automatically assigns to the new user both the role authenticated and the role all.

If you want to change the privileges of any of the built-in users and roles, you should carefully analyze the potential impact. To get a global view of all currently assigned privileges, use privilege reports (see *Privilege Reports*).
4.1.5  Rename Users and Roles

To rename a user:

1. Click **Administration**, and then click **Users**.
2. Click the user record you want to edit.
3. Enter the new name in the **User name** text box, and then click **Save**.

**Notes:**
- When a user name is changed, the currently assigned user password remains unchanged.
- If you are changing your own name (provided that you have this privilege), the changed name becomes effective as soon as you click Save, and is visible in the top right area of the page.

To rename a role:

1. Click **Administration**, and then click **Roles**.
2. Click the record you want to edit.
3. Enter the new role name in the **Role name** text box, and then click **Save**.

**Notes:**
- The members of a role do not change when the role is renamed.
- The default roles  all and authenticated cannot be changed.

4.1.6  Assign Roles to Users

To assign one or more roles to a user:

1. Click **Administration**, and then click **Users**.
2. In the list of users, click the record you want to edit.
3. Under **Roles available**, select the roles that must be assigned to the user, and then click **Assign**.

To revoke one or more roles from a user:

1. Click **Administration**, and then click **Users**.
2. In the list of users, click the record you want to edit.
3. Under **Roles assigned to user '<user name>'**, select the roles that must be revoked from the user, and then click **Remove**.
To assign a role to multiple users:

1. Click Administration, and then click Roles.
2. In the list of roles, click the record you want to edit.
3. Under Users/Roles available, select the users that must be assigned the role, and then click Assign.

To revoke a role from multiple users:

1. Click Administration, and then click Roles.
2. In the list of roles, click the record you want to edit.
3. Under Members of role '<role name>', select the users from whom the role must be revoked, and then click Remove.

### 4.1.7 Assign Roles to Other Roles

You can model the hierarchy of your organization or business within FlowForce Server by assigning roles to other roles. For example, you can create a role called 🧑‍💼 Employees and a role called 🧑‍💼 Marketing Department. Then you can assign the role 🧑‍💼 Marketing Department to be a member of 🧑‍💼 Employees. This means that all privileges and permissions granted to 🧑‍💼 Employees will be automatically inherited by users who are members of 🧑‍💼 Marketing Department.

To assign a role to another role:

1. Click Administration, and then click Roles.
2. In the list of roles, click the role you want to assign to another role (for example, if you want the role 🧑‍💼 Marketing Department to inherit privileges from the role 🧑‍💼 Employees, click "Employees").
3. Under Users/Roles available, select the role to be assigned, and then click Assign.

See also

- How Privileges Work
- How Permissions Work

### 4.1.8 Reset the Root Password

In the event that you forgot or lost the password of the 🧑‍♂️ root user account, you can reset it to the default value from the command line interface (see the command resetpassword).

To perform root password reset, it is assumed that you have access to the operating system where FlowForce is running, including FlowForce binaries and data files. This is the same kind of access required when installing FlowForce or when migrating to a new FlowForce version or server manually.

When you perform a password reset, the privileges of the 🧑‍♂️ root user will also be restored to the default value (that is, all the privileges will be granted).
Performing a root password reset does not affect any FlowForce users except the root user.
4.2 Privileges

This section includes the following topics:

- How Privileges Work
- Viewing Privilege Reports

4.2.1 How Privileges Work

Privileges define what users can do in FlowForce Server (for example, set own password, read users and roles, stop any job, and so on). Privileges are different from permissions in the sense that permissions control user access to containers, whereas privileges are effective globally across FlowForce Server. The following simple rule might help you distinguish quickly between privileges and permissions: privileges are global, permissions are local.

Like permissions, privileges can be assigned both to individual users and to roles. Therefore, when users log on to FlowForce Server, their set of effective privileges is determined by:

a) the privileges they have been assigned directly
b) the privileges assigned to any roles that the user is member of.

The following privileges are available in FlowForce Server.

<table>
<thead>
<tr>
<th>Define execution queues</th>
<th>Grants rights to define settings pertaining to job execution queues, see Defining Queue Settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintain global settings</td>
<td>This privilege grants rights to change the FlowForce Server global settings available in the Settings page—that is, the time zone and the mail server settings. This is an administrative privilege and should only be granted to FlowForce Server administrators.</td>
</tr>
</tbody>
</table>
| Maintain users, roles and privileges | This privilege grants rights to add, edit, and delete the following data:
  - Users
  - Roles
  - Privileges
  - Passwords

This is an administrative privilege and should only be granted to FlowForce Server administrators. By default, only the user root has this privilege.

| Override security | Users with this privilege can change container permissions without having "write" security permission. This allows FlowForce Server administrators to regain access to resources accidentally rendered inaccessible.

This is an administrative privilege and should only be assigned to FlowForce Server administrators. By default, only the user root has this privilege. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Read users and roles</td>
<td>By default, users can see only their own user account and any roles they are member of. When granted this privilege, users can see all existing users</td>
</tr>
<tr>
<td>Privilege</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>and roles.</td>
<td>By default, only the user root has this privilege.</td>
</tr>
<tr>
<td><strong>Retrieve sensitive data</strong></td>
<td>This privilege grants the right to retrieve and view passwords as plain text. By default, only the user root has this privilege. This privilege should normally be reserved to root only, unless you have a good reason to do otherwise.</td>
</tr>
<tr>
<td><strong>Set own password</strong></td>
<td>This privilege grants users the right to change their own password. Users who do not have this privilege need to have their password set by a FlowForce Server administrator.                                                                                                        By default, the authenticated role, and hence every user account except anonymous, has this privilege.</td>
</tr>
<tr>
<td><strong>Stop any job</strong></td>
<td>This privilege grants the right to stop any running FlowForce Server job, regardless of the user who created it.</td>
</tr>
<tr>
<td><strong>View unfiltered log</strong></td>
<td>By default, users can see log entries related to configurations to which they have &quot;read&quot; access. If granted this privilege, users can read all log entries, including those not associated with a specific configuration.                                                                                             By default, only the user root has this privilege.</td>
</tr>
</tbody>
</table>

**Inheritance**

You can assign privileges either directly to a user (for example, to Alethia Alonso), or to a particular role (for example, to Marketing Manager). The latter approach is recommended, because it simplifies management of privileges in the long term. For example, users may switch departments, or they might join or leave your organization. In either case, maintaining privileges for each individual user may become a counter-productive task. By assigning privileges to roles rather than users, you decrease granularity, simplify maintenance, and focus on the business need of each group or department rather than on individual users.

You can model the hierarchy of your organization or business within FlowForce Server by assigning roles to other roles. For example, you can create a role called Employees and a role called Marketing Department. Then you can assign the role Marketing Department to be a member of Employees. This means that all privileges and permissions granted to Employees will be automatically inherited by users who are members of Marketing Department.

Furthermore, you can assign the Marketing Manager role to be a member of Marketing Department role. In this case, the Marketing Manager role will inherit privileges both from the Marketing Department and from the Employees roles. When a new marketing manager joins your organization, Alethia Alonso, if she is assigned the Marketing Manager role, she will inherit all other privileges from the broader roles.
As the diagram shows, [Alethia Alonso] inherits permissions and privileges from the role [Marketing Manager]. This role, in its turn, inherits privileges from the [Marketing Department], and so on.

In a newly installed FlowForce Server system, considering the default users and roles, the users and privileges diagram looks as follows.

As the diagram shows, every user in the system inherits the privileges defined in the [all] role. However, only existing users (in this case, [root]) inherit the privileges defined in the [authenticated] role. If you add any new users to FlowForce Server, they are automatically assigned to the [all] and [authenticated] role (and thus granted the privileges defined in those roles, if any), as follows.
As a FlowForce Server administrator, you might find it difficult to keep track of privileges assigned to each and every role or user, especially when the number of users and roles increases. To help you get a quick overview of all privileges currently assigned to users and roles, FlowForce Server provides the following reports:

- Privileges Report
- Privileges by User Report

To view these reports, click Administration, and then click Reports.

Privileges Report
This report lists the FlowForce Server privileges. For each privilege, you can see the users who have been granted that privilege or inherited it by virtue of their roles.
**Privileges by User Report**

This report lists the FlowForce Server users. For each user, you can see the currently assigned privileges, and whether they have been granted or inherited.
### Privileges by User Report

<table>
<thead>
<tr>
<th>User</th>
<th>Privilege Description</th>
<th>Source Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alethia Alonso</td>
<td>Read users and roles</td>
<td>inherited from Manager</td>
</tr>
<tr>
<td></td>
<td>Set own password</td>
<td>inherited from all, authenticated</td>
</tr>
<tr>
<td></td>
<td>Stop any job</td>
<td>inherited from Manager</td>
</tr>
<tr>
<td></td>
<td>View unfiltered log</td>
<td>inherited from Manager</td>
</tr>
<tr>
<td>Employee</td>
<td>Set own password</td>
<td>granted to Employee</td>
</tr>
<tr>
<td>Klaus Mayer</td>
<td>Set own password</td>
<td>inherited from all, authenticated, Employee</td>
</tr>
<tr>
<td>Manager</td>
<td>Read users and roles</td>
<td>granted to Manager</td>
</tr>
<tr>
<td></td>
<td>Stop any job</td>
<td>granted to Manager</td>
</tr>
<tr>
<td></td>
<td>View unfiltered log</td>
<td>granted to Manager</td>
</tr>
<tr>
<td>Natsumo Shinohara</td>
<td>Set own password</td>
<td>inherited from all, authenticated, Employee</td>
</tr>
<tr>
<td>all</td>
<td>Set own password</td>
<td>granted to all</td>
</tr>
<tr>
<td>anonymous</td>
<td>Set own password</td>
<td>inherited from all</td>
</tr>
<tr>
<td>authenticated</td>
<td>Set own password</td>
<td>granted to authenticated</td>
</tr>
<tr>
<td>root</td>
<td>Maintain global settings</td>
<td>granted to root</td>
</tr>
<tr>
<td></td>
<td>Maintain users, roles and privileges</td>
<td>granted to root</td>
</tr>
<tr>
<td></td>
<td>Override security</td>
<td>granted to root</td>
</tr>
<tr>
<td></td>
<td>Read users and roles</td>
<td>granted to root</td>
</tr>
<tr>
<td></td>
<td>Set own password</td>
<td>granted to root and inherited from all, authenticated</td>
</tr>
<tr>
<td></td>
<td>Stop any job</td>
<td>granted to root</td>
</tr>
<tr>
<td></td>
<td>View unfiltered log</td>
<td>granted to root</td>
</tr>
</tbody>
</table>
4.3 Permissions and Containers

This section includes the following topics:

- How Permissions Work
- Understanding Containers
- Creating, Renaming, and Moving Containers
- Viewing Container Permissions
- Changing Container Permissions
- Restricting Access to the /public Container

4.3.1 How Permissions Work

Permissions control user access to containers. Like privileges, permissions can be granted both to users and to roles. Therefore, if a user is a part of a role, any permissions granted to the role will automatically apply to the user as well.

By default, permissions set on a container are inherited from the parent container. For example, let's assume that container A has a child container B. Users who have permission to access container A will have by default permission to access container B as well. However, an administrator can redefine the permissions of any user or role at every level of the container hierarchy.

FlowForce checks container permissions when users interact with containers. For example, users can view or change the contents of a container only if they have been granted the required permissions. Permissions are not evaluated upon job execution; therefore, any permission changes will not apply retroactively to existing jobs.

For each FlowForce Server container, you can set the following permission types.

**Container**

The “Container” permissions define what users can do with objects in the current container.

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same access rights to this container as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Grants the user rights to list the contents of the container.</td>
</tr>
<tr>
<td>Read, Write</td>
<td>Grants the user rights to list the contents of the container and to create or delete objects in the container.</td>
</tr>
</tbody>
</table>

**Note:** To successfully create a new configuration object, or delete an existing one, users must be granted both the Container - Read, Write permission and the Configuration - Read, Write permission.

| No access       | Denies the user the right to enter the container (more specifically, the container appears to the user as disabled). |
**Configuration**

The "Configuration" permissions define what a user can do with configuration objects (namely, jobs and credentials) in the current container.

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same configuration object–related rights as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Grants the user rights to view details about configuration objects within the container (such as the execution steps or triggers of a job).</td>
</tr>
<tr>
<td>Read, Write</td>
<td>Grants the user rights to modify any configuration object within the container (for example, edit the trigger of a job).</td>
</tr>
<tr>
<td>Note:</td>
<td>To successfully create a new configuration object, or delete an existing one, users must be granted both the <strong>Container - Read, Write</strong> permission and the <strong>Configuration - Read, Write</strong> permission.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user the right to view the details of any configuration objects within the container (more specifically, configuration objects appear to the user as disabled).</td>
</tr>
</tbody>
</table>

**Credential**

This permission defines what a user can do with **Credentials** defined in this container.

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same credential–related rights as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>Grants the user rights to reuse any credentials defined in this container.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user the right to reuse credentials defined in this container.</td>
</tr>
</tbody>
</table>

**Queue**

This permission defines what a user can do with queues defined in this container.

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same queue rights as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>Grants the user rights to assign a job to any queue defined in this container.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user the right to assign a job to queues defined in this container.</td>
</tr>
</tbody>
</table>

**Service**

The "Service" permission defines access to a job exposed as a Web service, via the HTTP request interface.

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same service–related rights as those defined on the parent container.</th>
</tr>
</thead>
</table>

Manage User Access Permissions and Containers

Use
Grants the user rights to access the service and thus execute the job via the request interface.

Notes
- Service permission checks skip any container hierarchy checks. Therefore, if granted Use permission, users may use the service without having Read access to the container in which the corresponding job is defined.
- If you grant Use permission to user anonymous, the service becomes publicly available and does not require authentication.

No access
Denies the user the right to access the job as a Web service.

Function
In addition to jobs, credentials, and other configuration data, a container may contain functions. These include built-in FlowForce functions, RaptorXML functions, and MapForce mappings or StyleVision transformations deployed to FlowForce.

When a FlowForce user creates a job, some execution step in their job may refer to functions from the same container, or from a different one. The "Function" permission defines whether users can invoke (refer to) functions from the container where the permission is defined.

For example, let's assume that an administrator has deployed various MapForce mappings to a FlowForce container called “Restricted”. The administrator can then decide if users should be able to refer to functions in this container, by changing the "Function" permission. More specifically, any user or role who has the Function - Use permission on container "Restricted" can refer to functions from this container (i.e., select them from a drop-down list when they create an execution step). On the contrary, users or roles with the Function - No Access permission will not be able to select any function from the "Restricted" container.

If an administrator revokes users’ access to functions after they had already used the function in a job, those users won't be able to run the job any longer. The job configuration page displays in this case a message with the text "You don't have permission to use the selected function".

Inherit
Provides to the user the same function-related rights as those defined on the parent container.

Use
Grants the user rights to call (refer to) any function defined inside the container.

No access
Denies the user rights to call (refer to) any function defined inside the container.

Resources
This permission defines what a user can do with Resources defined in this container.

Inherit
Provides to the user the same resource-related rights as those defined on the parent container.

Use
Grants the user rights to reuse (refer to) any resources defined in this container.

No access
Denies the user the right to reuse (refer to) any resources defined in this container.
Security
The security permission controls access to permissions of any child containers defined in the current container.

By default, users are permitted to read only permissions applicable to them (that is, any permissions assigned to themselves or any role they are a member of). However, users who have the Read users and roles privilege can read all permission entries.

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same security–related rights as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Security</td>
<td>Grants the user rights to view the permissions of any child of the container.</td>
</tr>
<tr>
<td>Read and Write Security</td>
<td>Grants the user rights to change the permissions of any child of the container.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user rights to view the permissions of any child of the container.</td>
</tr>
</tbody>
</table>

4.3.2 Understanding Containers

As the name of the term implies, a container is data packaged together. In FlowForce Server, containers can be roughly compared to folders on an operating system. Containers can contain any of the following: jobs, credentials, functions, and other containers. By setting permissions on containers, you can control who can view or access the data inside them. Organizing data into containers and setting up the relevant permissions for each container is a good security practice.

The top-level container in FlowForce Server is the root (/) container. By default, the root container contains the following predefined FlowForce Server containers.

| /public | The /public container is the default location where any FlowForce user can create jobs and credentials. It is by default empty and accessible to any FlowForce user. The /public container serves as default location in the following cases:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• When you deploy mappings from MapForce to FlowForce Server.</td>
</tr>
<tr>
<td></td>
<td>• When you deploy transformations from StyleVision to FlowForce Server.</td>
</tr>
<tr>
<td></td>
<td>You can, however, deploy mappings or transformations to a different container, if required.</td>
</tr>
<tr>
<td>/RaptorXML</td>
<td>This container is present if you licensed RaptorXML Server. It stores the validation and other functions specific to RaptorXML Server.</td>
</tr>
<tr>
<td>/system</td>
<td>The system container contains the FlowForce Server system functions. It is not recommended to make changes to this container.</td>
</tr>
</tbody>
</table>
You can navigate through containers from the Web administration interface, by clicking on a container to view its contents. The following screen shot shows a sample /public container that contains several configuration objects.

Sample FlowForce container

To go back to any container in the hierarchy, use the breadcrumb-style navigation available at the top of the page.

You can also search objects either within the current container including children objects (if the Recursive check box is checked) or only within the current container (if the Recursive check box is unchecked).

Containers contain objects such as jobs, deployed MapForce mappings or StyleVision transformations, functions, credentials. When you open a container, the following information is available about its objects:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the name of the object on the file system. Note that, when you create a new object, the name must not be already in use.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the object type (such as credential, job, or function). You can also identify the object type by its accompanying icon:</td>
</tr>
<tr>
<td>Date modified</td>
<td>Specifies the date and time when the object was created or last modified.</td>
</tr>
</tbody>
</table>

- Credential
- Function (includes built-in functions, MapForce mappings and StyleVision transformations)
- Job
- Container
- Missing configuration object. You may see this icon when you attempt to import into FlowForce Server data that has unresolved dependencies, see Handling Missing Dependencies.
Provided you have permissions to do so, you can create any number of additional containers to store your custom FlowForce server data (for example, one for each department). Alternatively, you can store data in the /public container, which by default is available to any authenticated user. If necessary, it is possible to restrict access to the /public container (see Restricting Access to the /public Container).

You can also move, rename, and delete any containers where you have the relevant permissions.

## 4.3.3 Creating, Renaming, and Moving Containers

You can create, rename and move containers if you (or any roles you are member of) have the Container / Read, Write permission (see also How Permissions Work).

**Note:** It is not recommended to modify the contents of the /RaptorXML and /system containers, which are provided by FlowForce Server by default.

To create a container:

1. Click **Configuration**.
2. Click an existing container under which you want to create a new container. If you want to create the container at the top level of the hierarchy, omit this step.
3. Click the **Create Container** button located in the lower left part of the page.
4. Enter the name of the container. The following name restrictions apply:
   - It must not be empty
   - It must not begin or end with space characters
   - It can contain letters, digits, single space, underscore (_), dash (-), and full stop (.) characters.

5. Click **Save**.

To rename a container:

1. Click **Configuration**, and then navigate to the container you want to rename.
2. Select the check box next to the container, and click **Move or Rename Selected Object**.

3. Enter the name of the container in the Name box, and then click **Rename**.
To move a container:

1. Click Configuration, and then navigate to the container you want to move.
2. Click the Move or Rename Selected Objects button located in the lower left part of the page.

![Move or rename to another container](image)

3. Select the container's destination by doing one of the following:
   - Enter the path in the Container text box.
   - Use the interactive navigation controls to reach the destination container.
4. Optionally, set the new name of the container by typing it in the Name box.
5. Click Move.

To move multiple containers:

- Click the check boxes next to them, and then follow the same logic as for moving a single container.

To select or deselect all objects in the container:

- Click the topmost check box.

### 4.3.4 Viewing Container Permissions

You can view the permissions of containers where you have the relevant permissions to do so (see also [How Permissions Work](#)). By default, you can see your own permissions with respect to the container. If you are member of any role, you can also see the permissions available to roles of which you are member. If you have
the privilege *Read users and roles*, you can also see the permission of other users and roles with respect to the container.

**To view the permissions of a container:**

1. Click **Configuration**.
2. Do one of the following:
   - Click the **Permissions** button adjacent to the container record.
   - Enter the container, and then click the **Permissions** button available in the lower right corner of the page.

The **User and Role name** column displays any users and roles whose permissions you have rights to see. The **Permissions** column displays what permission types are available to this particular user or role with respect to the container. For example, the image below illustrates the default permissions available to role **authenticated** for the root (/) container.

![Permissions for /](image)

For the description of each permission type, see **How Permissions Work**.

**4.3.5 Setting Container Permissions**

You can change permissions of containers where the following is true:
- You (or any roles you are member of) have the Security / Read and Write Security permission on the parent container relative to the one where you want to change permissions. For example, to change the permission of container "Jobs" which is a child of container "Marketing", you must have the permission Security / Read and Write Security on container "Marketing" (see How Permissions Work).
- You (or any roles you are member of) have been granted the privilege Override Security (see How Privileges Work).

To change the permissions of a container:

1. Click Configuration.
2. Do one of the following:
   - Click the Permissions button adjacent to the container record.
   - Enter the container, and then click the Permissions button available in the lower right corner of the page.

3. Do one of the following:
   - To change the permissions of any of the listed users and roles, click the Change button next to the relevant user or role.
   - To add permissions for any users and roles that are not listed, click Add Permissions.
4. In the Edit Permissions section, search for the user or role whose permissions you want to change, and select the check box next to it. You can either search for users created in FlowForce Server, or, if Directory Service is enabled, for domain users. For more information about importing domain users into FlowForce Server, see Importing Domain Users and Roles.

5. Change each relevant group of permissions as required. For the description of each permission type, see How Permissions Work. If you want to modify all permission types with a single click, use the Inherit, Full access, and No access buttons.
6. Click Save Changes.
4.3.6 Restricting Access to the /public Container

The /public container (located under the top-level root container) is available by default in FlowForce Server. It acts as a location accessible to any FlowForce Server user and a location where any FlowForce Server user can store their data, without any predefined permissions. Therefore, by default, the /public container has the following permissions.

### Permissions for /public

<table>
<thead>
<tr>
<th>User or Role name</th>
<th>Permissions</th>
<th>Inherited From</th>
</tr>
</thead>
<tbody>
<tr>
<td>authenticated</td>
<td>Container: Read, Write</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Configuration: Read, Write</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Credential: Use</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Queue: Use</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Service: Use</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Function: Use</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Security: Read</td>
<td>/</td>
</tr>
</tbody>
</table>

This means that, by default, any FlowForce Server user who is member of the authenticated role can do the following:

- Add, modify, and delete objects inside the /public container (namely, jobs, credentials, or other containers)
- Reuse any credentials available in the /public container
- Access as a Web service any job located in the /public container, provided that the job was configured to be available as a Web service
- Refer to any function available in the /public container
- Read the permissions assigned to the /public container

**Note:** These permissions may also be inherited by any containers that are children of the /public container. Normally, any new container inherits the permissions of the parent container; however, permissions may have been overridden by the root user, or by other users with relevant privileges.
You can restrict access to the `/public` container, if required. Note, however, that the job configuration examples included in this documentation assume the existence of the `/public` container.

**To restrict access to the `/public` container:**

1. Revoke permissions on this container from the `authenticated` role (see Setting Container Permissions).
2. Create a new role and assign this role to all users who require permissions to the `/public` container (see Creating Roles and Assigning Roles to Users).
3. Assign to the new role only the required permissions (again, see Setting Container Permissions).
4.4  Password Policies

This section includes the following topics:

- How Password Policies Work
- Creating and Assigning Password Policies

4.4.1  How Password Policies Work

FlowForce Servers uses password policies to help administrators manage the complexity of user passwords. A password policy is a set of minimum requirements that a user password must meet in order to be valid (for example, at least \(N\) characters long).

The password complexity rules that you can define within a password policy are as follows:

- The total minimum length of the password (that is, the password must be at least \(N\) characters long to be valid)
- The minimum number of letters that the password must contain
- The minimum number of digits that the password must contain

You can define as many password policies as required (provided that you have the Maintain users, roles and privilege privilege). Once you define password policies, you can assign them to FlowForce users. A user account can have one password policy at a time.

When the user requests a password change, the system checks if the new password meets the complexity requirements defined in the user’s password policy. If the password does not meet the complexity requirements defined in the password policy, the password change is denied, and the system displays a relevant message.

When an administrator changes the password of a user, FlowForce Server does not enforce the password policy. Also, if the password policy changes, any existing passwords remain unaffected. In the latter case, the password policy will be enforced when users attempt to change the existing password.

By default, FlowForce Server includes an empty password policy which does not enforce any password complexity rules. FlowForce Server implicitly assigns the default password policy to any user account that does not have a custom password policy. The default password policy cannot be changed.

See also

- Creating and Assigning Password Policies

4.4.2  Creating and Assigning Password Policies

To create a new password policy:

1. Click Administration, and then click Password Policies.
2. Click Create Policy.
3. Enter the required password policy rules, and then click Save. The list of current users becomes available under the defined policy.
4. Click to select the user records that must be assigned the new policy, and then click Assign.

See also
- How Password Policies Work
5  Job Configuration

In this section, you will learn to configure a job in FlowForce Server. You can find information about job configuration procedures in the following subsections:

- Credentials
- Create a Job
- Duplicate a Job
- Input Parameters
- Execution Steps
- Execution Result
- Triggers
- Jobs as Web Services
- Cache Job Results
- Queue Settings
- Windows Network Paths
- FTP Support
5.1 Credentials

A credential object stores authentication information. This is typically the combination of user name and password associated with a user account on the operating system where the FlowForce Server job runs, but it can also be a set of HTTP or FTP credentials.

Credentials are required by each job; they make it possible to run the job as a particular operating system user. Specifically, when you create a job in FlowForce Server, you must supply the credentials of the user account with which the job must be executed. Note that if the user account does not have sufficient rights on the operating system, the job cannot execute successfully. Credentials are also required when calling built-in FTP functions, where authorization to an FTP server is required. File watch triggers also require password credentials.

In FlowForce Server, you can define credentials either every time when you create a new job (referred to as local credentials), or as standalone (reusable) credential objects. In the latter case, when creating a job, you can refer to the credentials defined previously instead of entering them again. Standalone credentials are also convenient because you can update them easily in one place when they change. In other words, when you update a standalone credential, the change affects all jobs that use that credential reference.

**Notes:**
- Users can refer to credentials from jobs only if they have the relevant permissions granted. To make credentials from a specific container accessible to a user or to a role, administrators must grant the **Credentials - Use** permission to that user or role (see How Permissions Work).
- Because the clear text password needs to be sent to the operating system's login function, passwords are stored in a reversible encrypted form in the FlowForce Server database. The administrator should make sure to restrict access to the FlowForce Server's database file, see FlowForce Server Application Data.

If you have licensed MapForce and MapForce Server in order to run mappings as FlowForce Server jobs, you can create credential objects not only in FlowForce Server, but also in MapForce, at mapping design time. You can optionally deploy credentials created in MapForce to FlowForce Server, either together with the mapping where they belong, or as individual objects. A deployed credential does not necessarily have to store any sensitive data such as username and password (although it can, depending on your choice).

For information about creating credentials in MapForce and deploying them to FlowForce Server, refer to MapForce documentation (https://www.altova.com/documentation). For instructions about creating and using credentials in FlowForce Server, see Defining Credentials and Referring to Credentials from Jobs. For details about setting or overriding credentials in mapping jobs, see Credentials in Mapping Functions.

5.1.1 Define Credentials

You can define credentials as standalone objects that are reusable across multiple jobs.

**Prerequisites**
- Ensure that you have the **Container - Read, Write** and **Configuration - Read, Write** permissions granted on the container where you want to store the credentials.
Defining credentials

To create a credential object:

1. Navigate to the container where you want to store the credentials.
2. Click **Create | Create Credential**, and fill in the credential fields.

The credential fields are as follows.

<table>
<thead>
<tr>
<th>Credential name</th>
<th>Mandatory field. This is the name by which the credential is identified in FlowForce Server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential description</td>
<td>An optional description that provides more information about this credential.</td>
</tr>
<tr>
<td>User name</td>
<td>Mandatory field. The name of the user associated with this credential. For example, if the credential will be used to identify a user account on the Windows operating system, enter the Windows user account name. To specify a user name in a Windows domain, use the form <code>username@domain</code>. If the credential usage is for HTTP or FTP (see below), this may also be the HTTP or FTP user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Specifies the credential’s password. The password may be an empty string if the context where it will be used requires only the username without password.</td>
</tr>
<tr>
<td>Allow usage for HTTP</td>
<td>Select this check box if the credential will be referenced in jobs that call Web services which require basic HTTP authentication.</td>
</tr>
</tbody>
</table>
### 5.1.2 Refer to Credentials from Jobs

Assuming that you have been granted the required permissions to use a credential object, you can refer to it from various contexts where credentials are necessary, for example:

- You have created a credential that identifies a user account on the operating system where FlowForce Server runs (that is, the option **Allow usage for job execution** is enabled). You may subsequently refer to this credential from multiple jobs. This example is described below.
- You have created a credential that identifies an FTP username and password (that is, the option **Allow usage for FTP** is enabled). You may refer such a credential from any job that calls an FTP function.

The following example is illustrative of the common case where you need to refer to password credentials that identify a user account on the operating system where FlowForce Server runs:

1. Create a credential where the option **Allow usage for job execution** is enabled, as illustrated in Defining Credentials.
2. Create a new job or edit an existing one.
3. Under "Credential", click **Select existing credential**, and browse for the credential record defined previously.
If you have jobs that contain credential records defined locally, you can refer to them as if they were credentials objects themselves, for example:

In this case, the credentials of the embedded job (the one that has local credentials) will be used as credentials of the main job. Note that credentials are linked, not copied: if you change the locally defined credentials in the embedded job, they will be propagated to the main job as well.
5.2 Create a Job

This topic provides instructions on how to create jobs from the FlowForce Server Web administration interface. The instructions will help you understand the structure of jobs and their settings. However, not every step and setting listed below are applicable in real-life scenarios. Job configuration is a flexible process, which allows you to find more than one way to achieve the same result. To get an idea of various things you can do with jobs, see Job Configuration Examples.

Prerequisites

Make sure that you have the following permissions for the container in which you want to create a new job:

- Container: Read, Write
- Configuration: Read, Write

Create a job

Before creating a job, it might be a good idea to store the credentials of the operating system user account with which the job will be executed. For more information, see Credentials. If you intend to pass values between steps or between jobs, see FlowForce Expressions. To create a job, follow the instructions below:

1. Click Configuration and navigate to the container in which you want to create the job.
2. Click Create and select Create Job.
3. Enter a job name and, optionally, a job description.
4. If the job requires any values to be passed to it at runtime, create the required job input parameters. For details, see Input Parameters.
5. In the Execution Steps section, add steps of the job. Every job must have at least one step.
6. If the last step of the job returns a result, and if you intend to use the result in other jobs, select the return type in the Execution Result section.
7. If you want FlowForce Server to cache the returned result, specify caching preferences.
8. In the Triggers section, add a trigger (or triggers) that will fire the job. If the job is run as a Web service, adding a trigger is not necessary.
9. If you want to expose the job as a Web service, click the check box Make this job available via HTTP.
10. In the Credentials section, select an existing credential record or specify a local credential. For details, see Credentials.
11. If the job returns a result that you want to use in other jobs or expose as a Web service, define the job's cache settings.
12. Optionally, define the job's queue settings.
13. Click Save. FlowForce Server validates the entered information. If there are any fields that require your attention, FlowForce Server will highlight them in red.
5.3 Duplicate a Job

You can create copies of existing jobs when necessary. The copied job will have the same characteristics (execution steps, triggers, caching settings, and so on) as the original one. This will save you time, for example, when you need to quickly create a job using an existing one as a template.

To create a copy of an existing job:

1. Open the existing job, and then click the **Save As** button at the bottom of the page.
2. Enter the name of the new job, and click **Save As**.

![Save as dialog]

**Note:** If the credentials of the existing job are defined locally within the job, FlowForce will prompt you to enter the password again, for security reasons. If the credentials are defined as standalone credentials, this step is not necessary. For information about standalone versus local credentials, see [Credentials](#).

**Note:** If certain job components cause conflicts when the job is duplicated, FlowForce displays an error and does not duplicate the job. For example, if you attempt to duplicate a job containing a Web service, the service is already in use by the original job and cannot be duplicated. In this case, either change the URL of the Web service, or remove it completely.

The duplicated job is saved to the same container as the existing job. If you want to move it to a different container, you can do that as a two-step operation:

1. First, save an existing job with a new name, as described above. You can also create multiple copies of the same job without leaving the page.
2. Go to the parent container page, select one or more jobs, and then click **Move Selected Objects**.
5.4 Input Parameters

In the context of a FlowForce Server job, input parameters represent some arbitrary information supplied to the job based on which the job will execute in a particular way. Job input parameters are similar to function arguments in a programming language. Input parameters can be of various types (for example, file or directory references, text, numbers, Boolean values, and others). For some job types, FlowForce Server creates the input parameters automatically.

Notes:
- FlowForce Server automatically adds an input parameter called triggerfile to jobs that use file system or HTTP triggers (see also Managing Triggers). The triggerfile parameter contains the name of the file that activated the trigger and must not be deleted.
- If a job is using parameters and is exposed as a Web service, FlowForce Server expects the parameters to be passed by the calling application in the request (see also Exposing Jobs as Web Services).

An input parameter consists of the following fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Mandatory field. Specifies the name of the input parameter. Because you may need to refer to this parameter subsequently from any of the job's execution steps, use a name as descriptive as possible. The input parameter name must start with a letter and may contain only the following characters: a-z, A-Z, 0-9, and '_'.</td>
</tr>
<tr>
<td>Type</td>
<td>Mandatory field. Specifies the data type of the input parameter, which can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• String</td>
</tr>
<tr>
<td></td>
<td>• String as file</td>
</tr>
<tr>
<td></td>
<td>• String as directory</td>
</tr>
<tr>
<td></td>
<td>• String as file or directory</td>
</tr>
<tr>
<td></td>
<td>• Stream</td>
</tr>
<tr>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td>• Boolean</td>
</tr>
<tr>
<td>Default</td>
<td>Optional field. Specifies the default value of the parameter. This value will be used if no value is specified by the job caller at runtime.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional field. Describes the purpose of the parameter. This description becomes available as a tooltip next to the parameter name, when you use the current job as an execution step of another job.</td>
</tr>
</tbody>
</table>

Use the following buttons to manage parameters.
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add" /></td>
<td>Add parameter.</td>
</tr>
<tr>
<td><img src="image" alt="Delete" /></td>
<td>Delete parameter.</td>
</tr>
<tr>
<td><img src="image" alt="Duplicate" /></td>
<td>Duplicate parameter.</td>
</tr>
<tr>
<td><img src="image" alt="Move" /></td>
<td>Move parameter up or down.</td>
</tr>
<tr>
<td><img src="image" alt="Undo" /></td>
<td>Undo a previous delete action.</td>
</tr>
</tbody>
</table>
5.5 Execution Steps

In this section, you will learn to add and manage execution steps of a job. The section is organized into the following topics:

- **Add a Step**
- **Process Steps Sequentially**
- **Process Steps Conditionally**
- **Step Errors**
- **Retry on Error**
- **Error Handling Strategies**
- **Postponed Steps**
- **Data Types in Steps**

### 5.5.1 Add a Step

Steps define what the FlowForce Server job must actually do (for example, delete a file, execute a MapForce mapping, or send an email). In its simplest form, a step is just an operation with either failed or successful outcome, and it requires a function to be executed (see **Execution step**). However, there are also steps that provide means to execute other steps conditionally or in a loop (see **Choose step**, **Error/success Handling step**, and **For-Each step**). You can create as many steps as required within the same job, and you can set the order in which they must take place.

To add a step, click the button corresponding to the step type:

You can create the following step types.

<table>
<thead>
<tr>
<th>Execution step</th>
<th>A step of type &quot;Execution step&quot; lets you execute a specific FlowForce function. Available functions include the built-in functions delivered with FlowForce Server, deployed MapForce mappings or StyleVision transformations, and the execution steps of other jobs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose step</td>
<td>A step of type &quot;Choose step&quot; lets you define the conditions under which other job steps should be executed. Choose steps have the following structure:</td>
</tr>
<tr>
<td></td>
<td><strong>When</strong> (some expression)</td>
</tr>
<tr>
<td></td>
<td><strong>Execute</strong> (some step)</td>
</tr>
<tr>
<td></td>
<td><strong>Otherwise</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Execute</strong> (some other step)</td>
</tr>
<tr>
<td></td>
<td>Under each &quot;Choose step&quot;, you can nest other Choose steps (sub-conditions), for example:</td>
</tr>
</tbody>
</table>
When {expression}
  When {expression}
  Execute (step)
Otherwise
  Execute (step)
Otherwise
  Execute (step)

Any number of conditional steps can be defined.
Within any **When** / **Otherwise** pair, FlowForce Server executes only the condition that is true; the other condition is ignored.

For more information, see [Processing Steps Conditionally](#).

<table>
<thead>
<tr>
<th>For-Each step</th>
</tr>
</thead>
<tbody>
<tr>
<td>A step of type &quot;For-each step&quot; lets you iterate through a sequence (for example, a list of files within a directory) and repeat an execution step any number of times. &quot;For-Each&quot; steps has the following structure:</td>
</tr>
<tr>
<td>For each item in sequence {sequence expression}</td>
</tr>
<tr>
<td>Execute (step)</td>
</tr>
<tr>
<td>FlowForce executes the step until it finishes looping through all items of the sequence expression.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error/Success Handling step</th>
</tr>
</thead>
<tbody>
<tr>
<td>When a step of a job fails, FlowForce Server aborts the job. You can use steps of type &quot;Error/success handling step&quot; to perform clean-up actions before the job finally exits (see Handling Step Errors).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Postponed steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>In some cases, you might want to let the job return a result first, and execute certain steps only afterwards. Steps that are executed after the job returns a result are called postponed. This is particularly useful in job exposed as Web services. For details, see <a href="#">Postponed Steps</a>.</td>
</tr>
</tbody>
</table>

Use the following buttons to manage steps.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add step" /></td>
<td>Add step.</td>
</tr>
<tr>
<td><img src="image" alt="Delete step" /></td>
<td>Delete step.</td>
</tr>
<tr>
<td><img src="image" alt="Duplicate step" /></td>
<td>Duplicate step.</td>
</tr>
<tr>
<td><img src="image" alt="Move step up or down" /></td>
<td>Move step up or down.</td>
</tr>
<tr>
<td><img src="image" alt="Undo a previous delete action" /></td>
<td>Undo a previous delete action.</td>
</tr>
</tbody>
</table>
5.5.2 Process Steps Sequentially

The simplest way to run a FlowForce job is to add to it a single step of type "execution step". Such a step can be used to call a FlowForce built-in function, a shell command or script, a MapForce mapping, a StyleVision transformation, a RaptorXML function, and so on.

If you add multiple execution steps to a job, FlowForce will process them sequentially, starting with the first (topmost) step down to the last step. This rule also applies to any sub-steps that a step may have. The diagram below illustrates this scenario.

Sequential processing

By default, if FlowForce encounters an error, processing stops at the step where the error was encountered, and any subsequent steps are not executed.
Sequential processing error

Sometimes, you might not want to break the execution of the whole job if a step fails. In this case, you can configure the function called by the execution step not to stop the job execution. All subsequent steps will then process normally even if a step fails. Note that this behavior can be applied only to steps that call the following functions:

- All functions under `/system/filesystem`,
- All functions under `/system/ftp`,
- All functions under `/system/mail`,
- The `/system/shell/commandline` function.

For more information, refer to the description of each function (specifically, the "Abort on error" parameter).

Sequential processing is just one of the ways to process FlowForce jobs and may not always suit your needs. For more advanced processing, see:

- Processing Steps Conditionally
- Step Errors

### 5.5.3 Process Steps Conditionally

To process steps depending on some condition, add a step of type "Choose". Such steps consist of two parts: a "when" part and an "otherwise" part.
As shown in the image above, both the "When" part and the "Otherwise" part require an execution step. Also, the "when" part requires an expression. This expression dictates which of the two parts will be executed when the job runs. Namely, if the "When" expression evaluates to Boolean true, then Step C will be executed. Otherwise, step D will be executed. Therefore, the processing sequence of the abstract job illustrated above will be as follows (provided that there are no execution errors):

1. Step A
2. Step B
3. Step C or D, depending on the expression
4. Step E
5. Step F

For example, in the step illustrated below, FlowForce will execute the built-in delete function only if the expression "fileexists" evaluates to Boolean true.
Note that "fileexists" must have been declared in a previous step (or perhaps as an input parameter), for example:

Assign this step's result to fileexists

Otherwise, FlowForce would display a syntax validation error:

If you intend to pass the result of a "Choose" step to another step, or declare it to be of a particular type, ensure that each "When" and "Otherwise" condition produce the same return type. Otherwise, the return type of the "Choose" step is the common return type of all "When" and "Otherwise" branches, which might trigger validation errors.

For example, consider a job that checks the value of a numeric parameter temperature to tell you whether the weather is hot or normal. Such a job might look as follows:
In the job above, notice that **Declare return type as** is set to "string". If temperature is greater than a particular value (in this example, 35), a string expression is computed, namely "It is hot". Otherwise, the string expression "It is normal" is computed. So far, both the "When" and the "Otherwise" condition produce a string, so the job can be successfully saved and executed.

However, if we had left the "Otherwise" condition empty, the job would have triggered a validation error, as shown below.
The reason is that the expected return type of the job is "string" (as indicated by the **Declare return type as** setting). If the "Otherwise" branch is not defined, it computes to nothing. "Nothing" is not the same as a string; therefore, this triggers the validation error.

The jobs illustrated on this page use FlowForce expressions. For an introduction to FlowForce expressions, see [FlowForce Expressions](#).

### 5.5.4 Step Errors

When a step of a job fails, the job is considered failed as well. To perform some clean-up actions before the job finally exits (such as logging or sending email notifications), you can create steps of type "error/success"
The error/success handling steps allow you to protect the execution of one or more steps, referred to as a “protected block”. The following diagram illustrates the structure of a protected block.

When any of the steps within the protected block fail, error handlers control what happens before the job finally exits. Error handlers can be one of the following:

- **On Success** (runs if all steps in the protected block were successful)
- **On Error** (runs if any of the steps in the protected block were not successful)
- **On Retry** (this handler runs only if the protected block has the retry count option set to a value greater than 0; it runs if any of the steps in the protected block has failed and a retry is going to follow). For details, see the "Retry on error" section below.
- **Always** (runs regardless of whether the steps in the protected block were successful)

When the protected block finishes executing, FlowForce Server executes any defined handlers based on the outcome. For example, on the diagram shown above, the protected steps are Step A and Step B, and the error handling logic is as follows:

- If A fails, then A, C and E will be executed.
- If B fails, then A, B, C and E will be executed.
- If A and B are successful, then A, B, D and E will be executed.
In practice, it is not necessary (although it is possible) to define all three handler types for every job that you want to handle. The most common scenario is to define only On Error and Always handlers. For example, the image below illustrates a simple protected block with On Error and Always.

The first step runs a script from the C:\scripts directory by invoking the \system\shell\commandline function. The execution of this step is protected by two handlers: On Error and Always. The On Error handler will be triggered only if the execution of the first step fails. More specifically, if the first step fails, the error handling step sends an email that contains the ID of the failed job instance in the subject line. The Always handler is executed unconditionally, regardless of whether the first step was successful or not. This handler
logs a message by running a script from the `C:\scripts` directory. For a more detailed worked example similar to the one illustrated above, see Adding Error Handling to a Job.

### 5.5.5 Retry on Error

There may be cases when you would like to attempt to run a step again if its execution has failed for whatever reason. For example, if the job takes input data from an online file, there is a possibility that the network is down or the service is unavailable at the exact moment when the job runs. To make it possible to retry in case of an error, FlowForce Server provides the `retry count` option. By default, this option is 0, which means that the protected block will execute one time only and no retries will be attempted. If you set this option to a value greater than 0, there will be as many attempts to run the whole protected block again when an error is encountered. On each retry, FlowForce discards the results of the previous one and starts execution of the protected block from the beginning. If the maximum number of retries has been reached and the protected block still encounters an error, then the job is considered failed. The error is propagated to the outer block and reported as such only if the maximum number of attempts has been reached and the protected block execution is still unsuccessful.

Consider the following example:
The diagram above illustrates a job that consists of two protected steps (namely, steps A and B benefit from error handling). The **Set retry count** option is set to 2 (which means "attempt two more times if the first execution was unsuccessful").

Let's suppose that, when the job runs for the first time, step B fails. This means that steps A, B, F will be executed. No error is reported yet, because there are two more attempts left.

On the second attempt, B fails again. This time, steps A, B, D, and F will be executed. The only difference is that now the **On Retry** block is executed also.

Finally, on the third (and last) attempt, B does no longer encounter an error. This time, steps A, B, D, E, and F will be executed. The job outcome is considered as successful.

If step B had failed on the third attempt, then steps A, B, C, D, and F would have been executed. The job outcome would be considered as failed.

If you need to execute only step B each time when a retry takes place (not both A and B), enclose step B within its own protected block and set the **retry count** option at this level only. The outer protected block could then have the standard **Set retry count** = 0.

The number of retries of any given job instance is reported in the FlowForce log. In addition, if you need to get and process the number of retries at run-time, call the **retry-count** expression function—it returns the number of retries that took place for the innermost protected block surrounding it.

**To add an execution step that will be retried multiple times in case of an error:**

1. In the job configuration page, click **new Error/Success handling step**.

2. In the "Execute with error/success handling..." section, enter the number of retries.

3. Click the **button under the “Execute with error/success handling...” section and add the actual execution step that is to be retried in case of an error. In the example illustrated below, a job that uploads a file to some FTP server will be re-tried 3 times in case of an error. Note that this is a job defined outside of the current job and called like a function. The implementation details of the called job are not relevant here; any other execution step can be re-tried in a similar manner.
4. Click the button under the mandatory "On error" handler and add an execution step that should be called in case of an error. In this example, a job that writes to a .log file is called in case of an error. Again, this is an external job called like a function, so its implementation details are not shown. For a basic implementation of error logging, see Add Error Handling to a Job.

5. Optionally, add any of the "On Success", "On Retry", or "Always" blocks. To configure such additional handler types, click new error/success handler, select the handler type from the list, and then click to add the required steps under the respective handler.

5.5.6 Error Handling Strategies

A protected block may handle multiple steps. To handle multiple steps, simply add them one after the other inside a protected block. In terms of their structure, steps within a protected block are exactly like the standard, non-handled steps (for example, you can execute functions, embed FlowForce expressions, create
loops, and so on). In some cases, however, steps inside protected blocks may require special treatment, as discussed below.

The first thing to consider is that an exception handler can contain multiple execution steps. For example, one step generates a file, another one applies a transformation to it, and the third one sends it as an email. This is a valid configuration. Still, handlers with multiple execution steps add complexity and should be weighed out carefully, because an error may occur inside the error handler step itself, for example.

When multiple steps exist inside the same handler, they will run sequentially until all of them are executed, or until a step fails. Any steps after the failing step will not be executed. Nevertheless, if a handler fails, its outcome will be handled by an outer handler, if one exists.

To address this, you might want to limit the number of steps inside the handler, and thus the chance of error. The fewer the steps inside a handler, the higher the chances that the handler will run to completion. If any step is critical in the sense that subsequent ones depend on it, you could add, for example, a new error handler for this step specifically, and continue execution of dependent steps only on success.

Another thing to have in mind is that one cannot refer to the result of a step that's within a protected block, from the exception handler. The reason is that, if a step fails, the result of the protected block would be something undefined, and it is impossible to process an undefined result.

Therefore, if an execution step is placed within a protected block with an exception handler, it is not possible to access the result of that execution step within the exception handler. It doesn't matter if the exception handler is **On Success**, **OnError** or **Always**.

The restriction applies only for steps within the protected block. Results of steps that are outside the protected block can still be accessed within the exception handler. To understand this better, let's consider the following example job:
In the job illustrated above, each step has a result. For example, the first step has \texttt{result1}, the second step has \texttt{result2}, and so on. If you need to access the value of any of these results, note the following:

- Step 1 is outside of a protected block, so its result is accessible to all other subsequent steps. Namely, \texttt{result1} is accessible from step 2 or step 3, and also from the error handler.
- Steps 2 and 3 are inside a protected block, which makes their result accessible only from subsequent steps of the same protected block, but not from any handler. In other words, \texttt{result2} is accessible from step 3 but it is not accessible from the error handler. The \texttt{result3} is not accessible at all, because there are no other steps following it.
- Finally, the result of the error handler, \texttt{result_handler}, is not accessible to any other steps, because it is inside a handler and is the only and last step of the handler. Had there been more steps after it, then they could have consumed the \texttt{result_handler} result.
Having in mind the above, the design of the protected block should account for the visibility of results. The exact solution depends on the case. For example, if the result of a step inside an **On Error** block is important because it creates a file name, you can enclose it inside its own protected block (nested protected block) and call the `failed-step` function to get the erroneous output and still create the file. Although it does not identify the step that caused the error, this function returns a **result** type which includes error information if one occurred. A **result** represents the abstract result of running a shell command, a MapForce transformation, or a StyleVision stylesheet and can be handled not only when it produces the expected output, but also when an error occurs. This is best understood by example and is further discussed in *Adding Error Handling to a Job*.

### 5.5.7 Postponed Steps

A typical FlowForce job returns a result *only after all processing steps have finished*, assuming that no error was encountered. As long as there are running steps, the job must wait for them to finish before returning the result. For jobs exposed as Web services, this means that the HTTP transaction must be kept open for the entire duration of the job execution, which may take several minutes or even hours in some cases, depending on the volume of processed data.

To handle such cases gracefully, you can use a special kind of FlowForce step called a **postponed** step. Postponed steps do not prevent a job from returning a result right away, because their processing will take place only after the job has returned a result (hence the name “postponed”). When a job contains postponed steps, FlowForce will first process all non-postponed steps, then it will return the job result, and only afterwards proceed to running any postponed steps.

Even though a job with postponed steps might return a result early, the job is considered in progress until the execution of all postponed steps has completed. The job is considered to have finished successfully if all its postponed steps have finished successfully as well.

You can create postponed steps anywhere in the job where a step is allowed. To do this, create a new job or open an existing one, and then click the **new Postpone step** button, under *Execution steps*:

![Execution Steps](image)

This adds a new “Postpone” sequence to the job configuration page:

![Postpone](image)

Click inside of the “Postpone” sequence to create the actual step that is to be postponed. You can also create multiple postponed steps, which is useful if postponed steps depend on each other, like in the example below. You can also nest postponed sequences; however, the execution order might become non-trivial in that case, see "Advanced scenarios" below.
Remarks:

- A postponed sequence is just like another job within the current job, and it can have the same structure as a job. For example, it may include (or be part of) loops or error handling (protected) blocks.
- A job may contain multiple unrelated postponed sequences. This could be important for error handling: if an error occurs within a postponed sequence, the others will not be affected. For details, see the "Error handling" section below.
- All postponed sequences are executed after all non-postponed steps, from top to bottom. For special cases, see "Advanced scenarios" below.
- It is possible to postpone steps within called sub-jobs, which are executed after the main job returns a result.

Example: Postponed steps
The following example illustrates a possible usage of postponed steps. This job is a good candidate because it is exposed as a Web service, and can be invoked at any time by a client, including from the browser. For details about such jobs, see Exposing Jobs as Web Services.
Step A runs a time-consuming shell command that lists recursively all the directories and files within a large system directory. For this reason, step A was put inside a "Postpone" sequence. Step B takes the standard output (stdout) produced by A and writes it to a file. Step B depends on the output produced by A and, therefore, has to be part of the postponed sequence as well.

Step C informs callers of the service that the task has been submitted successfully. This step has been deliberately added as the last one, even though it will be executed the first when the job runs.

Namely, whenever the Web service is called, the steps above will run in the following sequence: C*, A, B. The reason is that A and B are postponed steps, so C is executed first. The asterisk marks the point where the job returns a result.
The advantage of this configuration is that the job returns a result immediately after running step C and the HTTP transaction can end, freeing up server resources for other requests. After returning the result of the job, FlowForce proceeds to running postponed steps A and B in the usual sequence.

As a confirmation, if you invoke the job above in your browser, you will notice that the message "Task has been submitted successfully" is displayed in the browser without any significant delay, while the job continues running until it creates the `output.txt` file (this may take from a few seconds to a few minutes, depending on the size of the listed directory). If neither A nor B fails, the output file will be created at the path `C:\FlowForce\Postponed\output.txt`.

**Note:** In this example, step C has to be the last one in the job because it produces the result to be sent to the browser when testing the Web service, namely, the string "Task has been submitted successfully". If you move step C to the very top, it is still executed first, and the postponed step B is still the last executed step of the job. However, this would change the job result and the browser would display some empty output similar to `[]`. The reason is that the result of a FlowForce job is always the result of the last executed step. Postponed steps do not have a return value but produce an empty sequence instead.

### Advanced scenarios

A job may contain multiple postponed steps, not necessarily one after the other, or multiple sequences of postponed steps. For example, let's suppose a job consists of the following steps, in this order:

```
A
postpone B
C
postpone D
```

The steps above will run in the following order: A, C*, B, D. As you see, non-postponed steps are executed first, followed by the postponed steps. The asterisk marks the point where the job returns a result.

You can also add postponed steps within conditions (choose steps). In this case, the postponed step will be run only if the respective "when" or "otherwise" branch is run as well.

```
when expression=true
{
    postpone A
    B
    C
}
otherwise
{
    postpone D
    E
    F
}
```

In the abstract job above, if the expression evaluates to `true`, the steps will run in the following order: B, C*, A. Otherwise, the run order will be: E, F*, D. The asterisk indicates where the job returns a result.

In `for each` steps, the postponed steps will be processed after all non-postponed steps, in the same order as the loop that they were part of.
for each item in list
{
    A
    postpone B
}

For example, assuming that the loop runs three times, then the steps above will in the following order: A1, A2, A3*, B1, B2, B3, where:

- The digit indicates the loop number
- The asterisk indicates where the job returns a result.

You can also nest postponed steps within other postponed steps. In this case, the rules are:

- Outer steps of the same depth are processed first
- Nested postponed steps will be executed only after their parent sequence has finished.

For example, let’s suppose that a job consists of the following nested postponed steps:

A
postpone
[
    B
    postpone C
    D
    postpone E
    F
]
G
postpone
[
    H
    postpone J
    K
    postpone L
    M
]
N

According to the rules above, the steps will run in the following order: A, G, N*, B, D, F, C, E, H, K, M, J, L. The asterisk marks the point where the job returns a result.

If you need to create and test advanced configurations like the one above, remember that you can always track the execution order of steps from the FlowForce log.

Error handling
As mentioned previously, postponed steps belong to a postponed sequence. A job may contain multiple postponed sequences at various places in the job. When an error occurs, the general rules are as follows:
If a postponed step encounters an error, it will be cancelled, along with any subsequent steps in the same postponed sequence, and the error will be logged. Note that this affects only the current sequence. If there are other postponed sequences in the same job, they will continue to run.

If a postponed step within a protected block encounters an error, this will generally cancel all postponed steps that are part of that block. This acts as a rollback and is necessary, considering that a protected block may have been configured to be retried multiple times in case of error, see Retry on Error.

To illustrate how the first rule above works, let's look at the following example:

In the job above, step D runs first because it is the only non-postponed step.

Apart from step D, there are two "Postpone" sequences: the first one contains steps A and B and the second one contains step C. Let's now suppose that step A fails. In this case, FlowForce Server cancels the entire "Postpone" sequence and so it cancels step B as well. The sequence containing step C is not affected by the error, and will continue to run.
5.5.8 Data Types in Steps

When you pass the result of one step to another step or job, you will most likely need to change the data type of the step result. For example, if you create a step that lists the contents of a directory, the return type is `result`. In order to be able to process this result in a particular way (for example, to send it an email, or to write it to a file), you need to convert it from `result` to `stream` or to `string`.

You can change the return type of a step (let’s call it `step 1`) as follows:

1. Type a value in the **Assign this step's result to** field of `step 1`. You will need to refer to this value later in order to access the result of the step.
2. Add a new execution step after `step 1` (let’s call it `step 2`).
3. Set the `system/compute` built-in function as execution function of `step 2`.
4. In the Expression field of `step 2`, type an expression which converts the value returned by `step 1` into the required data type. For example, if the value returned by `step 1` is a command line result that returns some output, and if you want to convert it to stream, use:

   \[
   \text{stdout(result)}
   \]

   Where `result` is the name you gave to the value returned by `step 1`.

   The exact expression to use depends on the result you want to achieve. FlowForce Server provides various expression functions which you can combine with operators and produce meaningful expressions for use in steps. To understand the available options, see [Step Result Functions](#).

**Example**

Let’s assume that you have created an execution step which lists the contents of a directory, as follows:

```
Execution Steps

1. Execute function `/system/shell/commandline`
   Parameters:
   - Command: `dir`
   - Working directory: `C:\`

2. Assign this step's result to `name` as `result`
```

At this stage, the output of the step is a `result`. To use this result in other steps (which could, for example, write the directory contents to a file, or send an email with its contents), you need a return value of type `stream`, not `result`.

**To convert the output of the directory listing step to "stream" data type:**

1. Declare the result of the step 1 as `Step1Output`. To do this, type `Step1Output` in the **Assign this step's result to** field.
2. Add a new execution step which calls the `system/compute` function.
3. Under Parameters, enter the following expression: `stdout(Step1Output)`, where `step1output` is the result of step 1.

At this stage your job output data type is stream, as intended. You can now create a new execution step that will send the result of the second step to an email address.

To send the directory listing to an email address:

1. Change the expression of step 2 from `stdout(Step1Output)` to `content(stdout(Step1Output))`. By doing this, you are converting the stream value into a string value.
2. Declare the result of step 2 as `Step2Output`. To do this, type `Step2Output` in the `Assign this step’s result to` field.
3. Add a new execution step (step 3), with the following settings (note that the mail server settings must be configured before you can use the built-in `/system/mail/send` function):

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the <code>/system/mail/send</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Enter the sender’s email address.</td>
</tr>
<tr>
<td>To</td>
<td>Enter the destination email address.</td>
</tr>
<tr>
<td>Subject</td>
<td>Enter the subject of the email, for example “Directory Listing”.</td>
</tr>
<tr>
<td>Body</td>
<td>Click <code>Set to</code>, and then select <code>Step2Output</code>.</td>
</tr>
</tbody>
</table>

At this stage, the execution steps should look as follows:
4. Create a timer trigger that will run the job either once or periodically (for example, every 60 minutes).

5. Click **Save**.
5.6 Execution Result

You may need to declare the data type returned by the job if you intend to use the result of the job in other jobs or if you want to cache the result of the job. When you declare the return type, keep in mind the following points:

- Declaring the return type is meaningful only for jobs that actually return a result.
- If you want to cache the result of a job, you must declare the return type.
- The return type of a job must be the same as the data type of the last step in the job. If the last step of your job returns a data type other than the one declared for the job, FlowForce Server returns an error (e.g., *Types string and result do not match*). When type matching errors occur, use expression functions to change the data type of the last step in the job to the data type declared as the job return type. For details, see Data Types in Steps.

Return type of a job
To define the return type of a job, take the following steps:

1. Create a new job or open an existing one for editing.
2. Select a return type in the Execution Result section (see screenshot below).

![Execution Result Screenshot](image)

Return types
The available return types are listed below.

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ignore/discard</em></td>
<td>This is the default return type. It instructs FlowForce Server to ignore or discard the result of the job provided there is one. Select this option if the job does not return a result, or if you do not need to process the returned result in any way.</td>
</tr>
<tr>
<td><em>string</em></td>
<td>Specifies that the data returned by the job represents a text value (string).</td>
</tr>
<tr>
<td><em>stream</em></td>
<td>Specifies that the data returned by the job represents a stream.</td>
</tr>
<tr>
<td><em>number</em></td>
<td>Specifies that the data returned by the job is a numeric value.</td>
</tr>
<tr>
<td><em>boolean</em></td>
<td>Specifies that the data returned by the job is a Boolean value.</td>
</tr>
</tbody>
</table>
5.7 Triggers

When you create a job, you must specify conditions (or criteria) that will trigger the job. These conditions are known as triggers. FlowForce Server monitors any defined triggers and executes the job whenever the trigger condition is met.

You can create multiple triggers for the same job and enable or disable any of the defined triggers. Whenever any of the enabled triggers fires, FlowForce Server executes all steps of the job. If you use triggers in jobs that have parameters, all parameters must have default values; otherwise, the job will not be executed. The following types of triggers are available in FlowForce Server:

- **Timer triggers** allow you to schedule jobs to start at a specific time and run for a specific time interval. Time triggers have flexible recurring options: For example, they can be set to run daily, weekly, on specific days of the week or month.

- **File system triggers** allow you to monitor a file or directory for changes (e.g., modified or newly added files). Note that deleted files cannot be monitored. You can flexibly configure the directory polling interval (e.g., every 60 seconds) and optionally set the start and expiry date of the trigger. You can also use wildcards to filter specific files of the directory.

- **HTTP triggers** allow you to monitor a URI (Uniform Resource Identifier) for changes. Specifically, you can poll for changes the Last-Modified and Content-MD5 HTTP header fields. You can flexibly configure the polling interval (e.g., every 60 seconds) and optionally set the start and expiry date of the trigger.

Add a trigger

To add a trigger, click the button corresponding to the trigger type you are interested in.

Manage a trigger

Use the buttons to the right of a trigger to manage the trigger (see below).
5.7.1 Timer Triggers

Time triggers allow you to schedule jobs to start at a specific time and run for a specific time interval. Time triggers have flexible recurring options: For example, they can be set to run daily, weekly, on specific days of the week or month. The screenshot below illustrates a sample timer trigger.

Timer trigger structure

Timer triggers have the following parameters: Run, Repeat, Start, Expires, Time Zone, and Enabled (see descriptions below).

Run
Defines whether the trigger should fire once or every $N$ number of days. The following options are available:

- Once;
- Daily;
- On days of week;
- On days of months;
- On days in weeks of months.

Repeat
Defines the Repeat options of the trigger. The repeat events occur on days specified in the Run drop-down list (see previous parameter). The every field defines the repeat frequency in minutes. The from and to fields define the time range between repeat events.
Start
Defines the trigger’s starting date and time. The start date and time entries are mandatory if you selected Once from the Run drop-down list. When you click in the date field, a pop-up calendar opens, which allows you to select the start date. You can also type in the date manually.

Expires
Defines the date and time when the trigger expires.

Time zone
Defines the time zone applicable to the start and expiry date and time. The default time zone is defined in the server administration settings.

Enabled
The enabled check box allows you to enable or disable the trigger. This option is useful when you create and test new jobs.

5.7.2 File System Triggers

File system triggers allow you to monitor a file or directory for changes (e.g., modified or newly added files). Note that deleted files cannot be monitored. You can configure the directory polling interval (e.g., every 60 seconds) and optionally set the start and expiry date of the trigger. You can also use wildcards to filter specific files of the directory. The screenshot below illustrates a sample file system trigger.

File system trigger structure
File system triggers have the following parameters: Check, Of file or directory, Polling interval, Wait N seconds for settle, Start, Expires, Time zone, and Enabled (see descriptions below).

Check
Specifies how the trigger should poll the directory or file. Valid options are listed below:

- Newly created: The trigger fires whenever any new files or directories are added to the specified directory. In terms of server load, this option requires the least server resources. When a new trigger is added and the job is saved, any existing files in that directory will be considered as newly created and the job will be executed for each. If a file is deleted and then added again later, the job will be executed for it again. Note that this will happen only if the polling interval has already elapsed since the deletion. The trigger also fires if a file has been renamed. This trigger does not fire if any files from the polled directory are subsequently modified. If you need such behavior, see Modified Date below.
- **Modified Date**: The trigger checks the last modification timestamp of all the specified files. If any dates have changed or a new file has been added or renamed, the trigger fires. This option takes slightly more resources from the server than the previous one.

- **Content**: This option computes and stores a hash code for the specified file. After the polling interval has passed, the hash code is recomputed and compared to the stored value. If there is a difference, the trigger fires. Note that this can place considerable load on the server. If any dates have changed or a new file has been added or renamed, the trigger also fires.

**Of file or directory**
Besides being able to choose any path, in which you would like to check changes, you can also use wildcards to specify directories for a file system trigger. You can specify a path like `C:\inbound\A*\B*`. FlowForce will scan all the subdirectories of `C:\inbound`. It will first scan its child directories starting with `A` and then scan all the child directories of `A` for directories/files starting with `B`.

**Polling interval**
Specifies the frequency in seconds, with which the directory will be polled. The default value is 60 seconds. The minimum value is 1 second.

**Wait N seconds for settle**
Defines the time in seconds that the server will wait before starting the next job.

**Start**
Defines the trigger's starting date and time. This is an optional field. When you click in the date field, a pop-up calendar opens, which allows you to select the start date. You can also type in the date manually.

**Expires**
Defines the date and time when the trigger expires.

**Time zone**
Defines the time zone applicable to the start and expiry date and time. The default time zone is defined in the server administration settings.

**Enabled**
The enabled check box allows you to enable or disable the trigger. This option is useful when you create and test new jobs.

**The triggerfile parameter**
Whenever you create a file system or HTTP trigger, FlowForce Server automatically adds a `triggerfile` input parameter to the job (see screenshot below). When the job runs, FlowForce Server sets this parameter to the file that triggers the job (file system triggers) and the name of the temporary file that contains the downloaded content of the polled URI (HTTP triggers).

### Job input parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>triggerfile</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

You can pass the value of the `triggerfile` parameter as an input value in any subsequent steps of the job.
This way, you can use or process the triggering file as required. By default, the `triggerfile` parameter contains the absolute path of the triggering file. To extract portions of the path, use the file path expression functions. See an example of a job that uses the `triggerfile` parameter in Creating a Directory Polling Job.

### HTTP Triggers

HTTP triggers allow you to monitor a URI (Uniform Resource Identifier) for changes. Specifically, you can poll for changes the Last-Modified and Content-MD5 HTTP header fields. You can configure the polling interval (e.g., every 60 seconds) and optionally set the start and expiry date of the trigger. The screenshot below illustrates a sample HTTP trigger.

#### HTTP trigger structure

HTTP triggers have the following parameters: **Check**, **Of URI**, **Polling interval**, **Wait N seconds for settle**, **Start**, **Expires**, **Time zone**, **Enabled** (see descriptions below).

- **Check**: Specifies how the trigger should poll the URI. The following options are available:
  - *HTTP Header Date* instructs the system to check the Last-Modified HTTP header. If the Last-Modified HTTP header is missing, the Content-MD5 header is checked (see next option).
  - *Content* instructs the system to check the optional Content-MD5 HTTP header. This is a 128-bit digest used as a message integrity check. If the MD5 header has changed after the polling interval has passed, the trigger fires. If the header is not provided by the server, the content is retrieved and hashed locally.

- **Of URI**: In this field, you need to specify the URI you would like to check for changes.

- **Polling interval**: Specifies the frequency in seconds, with which the URI will be polled.

- **Wait N seconds for settle**: Defines the time in seconds that the server will wait before starting the next job.

- **Start**: Defines the trigger's starting date and time. This is an optional field. When you click in the date field, a pop-up calendar opens, which allows you to select the start date. You can also type in the date manually.
Expires
Defines the date and time when the trigger expires.

Time zone
Defines the time zone applicable to the start and expiry date and time. The default time zone is defined in the server administration settings.

Enabled
The enabled check box allows you to enable or disable the trigger. This option is useful when you create and test new jobs.

The triggerfile parameter
Whenever you create a file system or HTTP trigger, FlowForce Server automatically adds a triggerfile input parameter to the job (see screenshot below). When the job runs, FlowForce Server sets this parameter to the file that triggers the job (file system triggers) and the name of the temporary file that contains the downloaded content of the polled URI (HTTP triggers).

You can pass the value of the triggerfile parameter as an input value in any subsequent steps of the job. This way, you can use or process the triggering file as required. By default, the triggerfile parameter contains the absolute path of the triggering file. To extract portions of the path, use the file path expression functions. See an example of a job that uses the triggerfile parameter in Creating a Directory Polling Job.
5.8 Jobs as Web Services

You can expose FlowForce Server jobs as Web services. Jobs exposed as Web services are primarily meant
to be accessed programmatically. For testing and debugging purposes, however, you can also invoke such
jobs from a browser.

To make a job available as a Web service:

1. Create a new job or open an existing one for editing.
2. Select the Make this job available via HTTP... check box.
3. Type the name of the Web service in the "myURL" text box.

When you save the job, FlowForce Server starts a listener service which accepts HTTP GET and POST
requests at the specified request URL specified. Jobs exposed as services remain active as long as FlowForce
server is running.

Note: The Call Web Service button is available only if you have set the Host name field of the
FlowForce Server service from the Setup page. Clicking this button invokes the Web service in a
new browser window. If you haven't configured a host name for FlowForce Server, the button is not
displayed, but you can still call the Web service by typing its URL manually in the browser's address
bar.

When the Web service is invoked, FlowForce Server runs the job execution steps specified and returns one of
the following:

- The first result file of the last step, if the job produces a result file;
- The standard output of the last step, if no result files are produced (this might be the case when you
  are working with command line output).

A valid result is returned with a HTTP 200 status, with the Content-Type header set according to the result.
The Content-Type header depends on the actual result. A MapForce mapping will result in text/xml if it has
XML output, or text/plain for text output. Standard output of other functions is also returned as text/plain.
The result is returned as the response body, without any embellishments.

Execution errors are reported as HTTP 5xx status with a generic error message. For further information, check
the FlowForce Server log.

For a step-by-step example of how to configure such a job, see Exposing a Job as a Web Service.

It is possible to configure FlowForce to return a result before all the job steps are executed. This is particularly
useful if the job invoked as a service takes a long time. The early result could be treated by the caller as a
confirmation that the task has been accepted by FlowForce Server for processing. For details, see Postponed
Steps.
To view all currently configured FlowForce Server Web services:

Do one of the following:

- In the FlowForce Web administration interface, go to **Home** and click **Show all active triggers and services**. See also [Active triggers and services](#).
- Access the following URL from your browser: http://[FlowForceServer][ServerPort]/service/*

**Note:** In the URL above, [FlowForceServer] and [ServerPort] refer to the network address and port where FlowForce Server is listening. By default, FlowForce Server runs on [http://localhost:4646](http://localhost:4646) (assuming you are accessing it from the same machine). The server name and port are as defined in the administration page, see [Defining the Network Settings](#).

**Web service parameters**

When you expose a job as Web service, all job parameters automatically become parameters for the service. If a job parameter does not have a default, it is mandatory and must be provided when invoking the service. Optional parameters take the default value if not provided in the request.

When the service is invoked, FlowForce Server verifies the parameters supplied in the request against those defined in the job. If parameter validation fails, FlowForce Server returns a 5xx HTTP status. In this case, FlowForce Server also displays an HTML parameter form, for debugging and testing purposes. You can use the HTML form to enter manually any parameters, for debugging and testing. For each parameter of type *Stream*, a **Browse** button becomes available in the page, and you can use it to upload the file required as parameter.

To display the testing HTML form unconditionally, supply the built-in parameter **showform** in the request (with any value).

To call a FlowForce Web service with parameters, a client can use one the following options:

1. For parameters of simple type such as strings or numbers, a client can supply them in the URL of the GET or POST request. For an example, see [Expose a Job as a Web Service](#).
2. In case of POST requests, a client can additionally provide parameters as **multipart/form-data**, or as **application/x-www-form-urlencoded**. If the parameter is of type "stream" in FlowForce, then the client must provide them (and FlowForce expects them) as **multipart/form-data**. For such parameters, the browser test HTML form displays a **Browse** button next to the corresponding parameter, as mentioned above.
3. The client call can also include arbitrary content in the body of the POST request (this specifically refers to content such as JSON or XML posted not as a parameter but as the body of the HTTP request). In order for this to be possible, the FlowForce job must contain a single parameter of type *stream*. If you need additional non-stream parameters, these must be supplied in the POST URL. However, only one parameter of type *stream* must be defined in FlowForce; other parameters must be of non-stream type. When these conditions are met, the request body will be treated as data for the *stream* parameter. No other configuration is required. For an example, see [Post JSON to FlowForce Web Service](#).

**Web service authentication**

By default, FlowForce Server uses HTTP Basic authentication to authenticate clients calling a Web service. User credentials are checked against the FlowForce Server user database (the same user name and password used to log on to FlowForce Server Web administration interface).
To make a Web service available without credentials, grant the Use Service permission to the default anonymous user (see also How Permissions Work).

You can still supply HTTP credentials when a service is available for anonymous use. The credentials are then checked against the FlowForce Server user database and the service execution is attributed to the authenticated user instead of user anonymous.

If you supply invalid credentials, the request interface returns an HTTP status of 401. If you did not supply credentials and service use has not been granted to anonymous on this service, the request interface also returns an HTTP status of 401.

If you supplied valid credentials, but the authenticated user is not granted Use Service permission on this service, the request interface will return an HTTP 4xx failure status. If you try accessing a service that does not exist, an HTTP 4xx failure status is returned.

Optionally, domain authentication can also be configured, in addition to HTTP basic authentication. For information about how to configure it, see Changing the Directory Service Settings. Once domain authentication has been configured, users will be able to access Web services exposed by FlowForce Server, provided that they supply a valid username and password for the respective domain. Importantly, for Active Directory, the username must contain the prefix NT/ and must include the domain name, for example: NT/john.doe@my.domain.com.

Queue settings

Service URL requests are a particular kind of trigger, and are therefore subject to the same queue constraints once the connection has been established. See Defining Queue Settings.
Configuring the maximum size of the HTTP request body

A default limit exists in FlowForce Server that establishes the maximum size of the HTTP request body, which is around 100 MB. When a caller posts HTTP requests to FlowForce jobs exposed as Web services and the HTTP request body exceeds this limit, FlowForce Server may return an error with the following text:

The entity sent with the request exceeds the maximum allowed bytes.

To accept requests of larger sizes:

1. Open the flowforce.ini file in a text editor.
2. Add the option `max_request_body_size` to the [Listen] or [ListenSSL] section and set it to the maximum number of bytes that should be allowed.

For example, in order to enable a maximum size of 500 MB, your flowforce.ini file could look like this:

```
[Listen]
active=1
host=0.0.0.0
port=4646
hostname=somehost.example.org
max_request_body_size=500000000
```

For more information about the .ini file, see Configuration File Reference.

Reconfiguring FlowForce Server pool threads

If you expect a large number of parallel HTTP service requests (for example, 20 or more at a time), it is possible to reconfigure the server for a slightly larger number of pool threads.

1. Open the flowforce.ini file in a text editor.
2. Add the option `thread_pool` to the [Listen] or [ListenSSL] section of the .ini file and set it to a value larger than 20.
3. Restart the service.

**Note:** It is a good idea to have two separate [Listen] sections, one for FlowForce Web Server (which doesn’t require that many pool threads) and the other for all other requests (on a different port, preferably). Otherwise, FlowForce Web Server will be competing with all the other HTTP requests for pool threads.
5.9 Cache Job Results

Caching reduces both the server load and the response time of jobs. It is particularly useful for jobs exposed as a Web service or for data-intensive jobs that require a long time to complete and increase the server load.

Caching the result of a job essentially means that FlowForce Server prepares and stores the job result in some internal repository (that is, the cache). If the job has parameters, the system creates a cache entry for every parameter combination, up to a limit that can be configured. When the job with cached result is called from another job (referred to as the consumer), FlowForce Server returns the cached result to the consumer (instead of executing the job again), thus reducing the response time.

The right balance between keeping the server load under control, on one hand, and not letting the cache become too old, on another hand, depends on the specifics of your environment and the jobs used. To achieve this balance, you can experiment with the caching settings provided by FlowForce Server and then decide what works best for you.

When working with cached job results, note the following:

- It is mandatory to declare the data type returned by the job whose result is cached (see Declaring the return type of a job).
- Both the job whose result is cached and the consumer job must use the same credentials. If the credentials differ, then the job executes as if no cache were defined.
- When you change the configuration of the cached job, the existing cache data is invalidated (flushed).

### Caching Result

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache the result: Check this box if you want the job results to be cached. By doing so, you are instructing any consumers of the current job to read the cached result rather than execute the job.</td>
</tr>
</tbody>
</table>

### Caching settings

The available caching settings are as follows.

<table>
<thead>
<tr>
<th>Cache the result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box if you want the job results to be cached. By doing so, you are instructing any consumers of the current job to read the cached result rather than execute the job.</td>
</tr>
</tbody>
</table>

If the current job is executed directly (not through a consumer), either because a defined trigger fired or because the job's Web service is invoked, FlowForce Server refreshes the cache (or, if the job parameters are not found in the cache, it creates a new cached entry based on the supplied parameter combination).
<table>
<thead>
<tr>
<th><strong>Initiated by consumer</strong></th>
<th>When this option is enabled, any job that is calling the current job (that is, the consumer job) will compute and populate the cache if it does not exist. Otherwise, only triggers and Web service calls will populate the cache.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maximum number of cache entries</strong></td>
<td>This option restricts the number of cached job results per job. When the job has parameters, you might want to set this option to the number of all possible parameter combinations (provided the combinations are finite).</td>
</tr>
<tr>
<td><strong>Auto create a new cache consumer job</strong></td>
<td>A cache consumer job is a Web service at the HTTP address you specify. The consumer Web service acts as a convenient way to retrieve and manage the cache of the job whose result is being cached. When invoked, the consumer job attempts to use the cached result of the main job in first place. If there is no cached result and the <strong>Initiated by consumer</strong> option is disabled, the consumer retrieves the actual result returned by the main job. If there is no cached result and the <strong>Initiated by consumer</strong> option is enabled, the consumer retrieves the actual result returned by the main job and also populates the cache.</td>
</tr>
<tr>
<td><strong>Refresh Cache timer</strong></td>
<td>A Refresh Cache timer controls how often the system should refresh the cache of the current job. All currently cached parameter combinations are refreshed.</td>
</tr>
<tr>
<td><strong>Purge Cache timer</strong></td>
<td>A Purge Cache timer controls how often the system should purge the cache of the current job.</td>
</tr>
</tbody>
</table>
Refresh Cache timers and Purge Cache timers can be deleted in the same way as other triggers, by clicking the **Delete** button. Also, the **Undo Delete** operation becomes available for them until you save or reload the page. The **Duplicate** button enables you to create a copy of the current trigger, with the same settings.

**See also**
- [Caching Job Results](#) (example)
5.10 Queue Settings

Queue settings enable you to control usage of server resources more efficiently. For example, through queue configuration, you can limit the number of job instances running in parallel at any given moment.

An execution queue is a “processor” of jobs; it controls how job instances run. In order to run, every job instance is assigned to a target execution queue. The queue controls how many job instances (of all the jobs assigned to the queue) can be running at any one time and the delay between runs. By default, the queue settings are local to the job, but you can also define queues as standalone objects shared by multiple jobs. When multiple jobs are assigned to the same execution queue, they will share that queue for executing.

Queues benefit from the same security access mechanism as other FlowForce Server configuration objects. Namely, a user must have the "Define execution queues" privilege in order to create queues, see also How Privileges Work. In addition, users can view queues, or assign jobs to queues, only if they have appropriate container permissions (not the same as privileges), see also How Permissions Work. By default, any authenticated user gets the "Queue - Use" permission, which means they can assign jobs to queues. To restrict access to queues, navigate to the container where the queue is defined, and change the permission of the container to “Queue - No access” for the role authenticated. Next, assign the permission “Queue - Use” to any specific roles or users that you need. For more information, see Restricting Access to the /public Container.

Creating standalone queues
To create a queue as a standalone object:

1. Click Configuration, and then navigate to the container where you want to create the queue.
2. Click Create, and then Create Queue.
3. Enter a queue name, and, optionally, a description. For reference to all settings, see “Queue settings” below.
4. Click Save.
Defining local queues

As an alternative to creating standalone queues, you can define the queue settings locally inside the job. To do this, select the Define local queue option from the job configuration page and then specify your queue preferences. The image below illustrates the default queue settings.

If you choose the Select existing queue option, you must specify a standalone, external queue defined previously. For reference to the Minimum time between runs and Maximum parallel runs settings, see the "Queue settings" section below.

Queue settings

The settings available for configuration in a queue are listed below.

| Queue name     | Enter a name that identifies the queue. This is a mandatory field, and it must not start or end with spaces. Also, it may contain only letters, digits, single spaces, and the underscore ("_"), dash ("-"), and full stop ("." ) characters.
|                | This field is applicable only if the queue is defined as standalone (not local) queue.

| Queue description | Optionally, enter a description for the queue object.
|                   | This field is applicable only if the queue is defined as a standalone (not local) queue.

| Minimum time between runs | An execution queue provides execution slots. Each slot will execute job instances sequentially.
|                          | The "Minimum time between runs" setting keeps a slot marked as occupied for a short duration after a job instance has finished, so it will not pick up the next job instance right away. This reduces maximum throughput for this execution queue, but provides CPU time for other execution queues and other processes on the same machine.

| Maximum parallel runs | This option defines the number of execution slots available on the queue. Each slot executes job instances sequentially, so the setting determines how many instances of the same job may be executed in parallel in the current queue. Note, however, that the number of instances you allow to run in parallel will compete over available machine resources. Increasing this
value could be acceptable for queues that process "lightweight" jobs that do not perform intensive I/O operations or need significant CPU time. The default setting 1 is the most conservative and is suitable for queues that process resource-intensive jobs (so as to ensure only one such "heavyweight" job instance is processed at a time).

This option does not affect the number of maximum parallel HTTP requests accepted by FlowForce Server (such as those from clients that invoke jobs exposed as Web services). For details, see Reconfiguring FlowForce Server pool threads.
5.11 Windows Network Paths

When you create jobs, you will typically need to refer to file paths on the machine where FlowForce Server runs or to file paths on the network. When you refer to a Windows network path (e.g., a mapped network drive), use the Universal Naming Convention (UNC) syntax. This is necessary because drive letters are not global to the system, and each logon session is assigned its own drive letters.

The UNC has the following syntax: `\\server\sharedfolder\filepath`, where `server` references the server name in the network (defined by the DNS); `sharedfolder` references a label defined by the administrator (e.g., `admin$` is generally the root directory of the operating system installation); `filepath` refers to the subdirectories below the share.
5.12 FTP Support

When configuring jobs in FlowForce Server, you may need to connect to an FTP server. For example, your job may need to retrieve files from the remote FTP server, list the contents of a directory on the FTP server, or delete files via FTP. To enable operations such as the one above, FlowForce Server provides support for connecting to FTP Servers using any of the following protocols: FTP, FTPS, SFTP. Information about these protocols is provided in the list below:

- The File Transfer Protocol (FTP) enables you to transfer files between FlowForce Server and the remote FTP server. This protocol is assigned the TCP port 21. The FTP protocol was not designed to be a secure mode for transferring files. If you would like to transfer data securely, use either FTPS or SFTP, as described below.

- An alternative to using plain FTP is the FTP Secure (FTPS) protocol, which refers to FTP over a secured TLS/SSL channel. FlowForce Server supports this usage of FTP according to RFC 4217 (also known as FTP with TLS/SSL explicit encryption). To establish a connection via FTP or FTPS, use functions available in the built-in /system/ftp container.

- FTP with TLS/SSL implicit encryption is considered deprecated and thus not supported.

- Secure FTP (SFTP) is a method of transferring data securely over Secure Shell (SSH). FTP over SSH works in a significantly different way compared to plain FTP.

- SFTP support is available only in FlowForce Server Advanced Edition.
6 Import/Export Configuration Data

You can export jobs and other configuration objects (including deployed MapForce mappings and StyleVision transformations) from FlowForce Server as follows:

- To another running FlowForce Server instance (online export)
- To a file (offline export)

When you export objects to another running FlowForce Server instance, the exported objects become immediately available in the Web administration interface of that server.

When you export objects to a file, FlowForce Server creates a .zip archive which contains the selected objects and their dependencies. The .zip archive is named according to the date and time when the export operation took place. The naming convention is `export_YYYYMMDDhhmmss`. For example, a file exported on the 6th of August 2015-2021 at 10:51:33 server time would be named `export_2015-20210806105133.zip`.

You can subsequently import the .zip archive either into the same FlowForce Server instance (provided the imported objects no longer exist at destination, or you want to overwrite them), or into another instance.
6.1 Exporting Configuration Data

You can export either specific records within a container, or entire containers. In either case, FlowForce displays a dialog box that enables you to review the list of records before exporting them. If you selected an entire container for export, this dialog box displays all the children records of the selected container (jobs or credentials).

Before exporting objects, FlowForce informs you on a separate page about all objects that are dependent on (or are referenced by) the objects that you wish to export. This helps you see at a glance if there are missed dependencies. If you are exporting objects to a running FlowForce Server, you can also see whether each object already exists on the destination server.

By default, FlowForce Server does not export the following categories of sensitive data:

- Passwords defined locally in jobs
- Passwords available as "standalone" credential records.
- Passwords stored with system functions (such as /system/ftp functions)

To export all these categories of sensitive data, select the option Export sensitive data during export. Be aware that, if you select the check box, the exported archive will include the sensitive data in plain text form.

If you do not select the check box, the sensitive data will not be exported. Upon importing data back into FlowForce Server, you have the option to overwrite each individual record, or skip it. If you choose to overwrite, the existing sensitive data will be replaced with empty values.

To export configuration data to a running FlowForce Server instance:

1. Click Configuration, and select the records you want to export. You can select either specific records within a container, or the entire container.
2. Click Export Selected Objects.
3. In the dialog box, click **Export to server**, and enter the host name of the destination FlowForce Server, and the port where it runs.

4. Enter your user name and password on the destination FlowForce Server instance, and then click **Export**. FlowForce displays all records to be exported on a page where you can view their dependencies, or omit them from the export.
The records with a yellow background are those that are being exported. The record without a yellow background are those that you have excluded from the export, by clearing their adjacent check box. Finally, the records that are grayed out represent dependencies on built-in system functions, so you cannot take actions on them.

The "Remote Server" column indicates if the file exists at destination. If the dependencies already exist at destination, you can safely omit such records from the export. Otherwise, if you are exporting without dependencies and the dependencies do not exist at destination, such jobs will likely fail, see also Worked Examples.

As mentioned above, the Export sensitive data option lets you optionally include passwords in the exported package. For security reasons, it is not recommended to select this check box unless you really need to transfer such sensitive data in plain text out of FlowForce Server.

The check box Downgrade credentials for FlowForce 2019r3 or earlier must be selected if the exported list includes records of type "credential" and if the target FlowForce Server is of version 2019r3 or earlier. After that release, credentials got new "Allow usage" options, and so the check box makes it possible to make newer credential records compatible with older versions of FlowForce. For more information about "Allow usage" options, see Defining Credentials.

5. Click to select the objects to be exported, and then click Start Export.

To export jobs to a file:

1. Click Configuration, and select the records you want to export. You can select either specific records within a container, or the entire container.
2. Click Export Selected Objects.
3. In the dialog box, click **Export to file**, and then click **Export**. FlowForce displays all records to be exported on a page where you can view their dependencies, or omit them from the export.
The records with a yellow background are those that are being exported. The record without a yellow background are those that you have excluded from the export, by clearing their adjacent check box. Finally, the records that are grayed out represent dependencies on built-in system functions, so you cannot take actions on them.

Because you are exporting to a file and not to a running FlowForce Server instance, it is not possible to determine whether the exported objects exist at destination. For this reason, the "Remote Server" column shows "Might or might not exist on remote server". If the dependencies will exist at destination when you import the .zip archive back into FlowForce, you can safely omit such records from the export. If you are not sure, choose to export all dependencies. Otherwise, when you later attempt to import data where dependencies are missing, the import will fail with an error like "Operation failed: Path does not exist". See also Worked Examples.

4. Click **Start Export to File**. Depending on your browser settings, you may either be prompted to save the .zip archive to a local directory, or the browser may save it automatically to a preconfigured destination directory.
6.2 Including or Excluding Sensitive Data

When you export data from FlowForce Server, you can choose whether to include or exclude sensitive data from the exported archive. The examples below explain the differences between the two approaches.

Example 1: Exclude sensitive data

Let’s assume that you have a job (“AddNumbers”) which refers to a credential record that is in the same container (“my.credentials”).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>my.credentials</td>
<td>credential</td>
</tr>
<tr>
<td>AddNumbers</td>
<td>job</td>
</tr>
</tbody>
</table>

If you choose to export both objects to a file (.zip archive) without selecting the Export sensitive data option, the following happens:

1. The job will be exported.
2. The credential record will be exported without the password.

If you later import the .zip archive into a FlowForce Server environment where the two objects do not exist, both objects will be created successfully. Note that the password associated with the credential record will be empty.

If the objects already exist in the target environment, you can overwrite them or clear the corresponding check box and skip them:

If you choose to overwrite both records, the following happens:

1. The job existing in FlowForce Server will be overwritten by the job from the .zip archive.
2. The credential record existing in FlowForce Server will be overwritten by the one from .zip archive, and the destination password will become empty.

If you do not overwrite the credential, the existing credential remains untouched.
Example 2: Include sensitive data
Let's assume that you export the same two records as above, and also select the Export sensitive data option during export. In this case, the following happens:

1. The job is exported
2. The credential record is exported and includes the password as well.

If you later import the .zip archive into a FlowForce Server environment where the two objects do not exist, both objects will be created successfully. The password associated with the credential record will be the one from the .zip file.

If the objects already exist in the target environment, you can overwrite them or clear the corresponding check box and skip them. If you choose to overwrite the records, the following happens:

1. The job existing in FlowForce Server will be overwritten by the job from the .zip archive.
2. The credential record existing in FlowForce Server will be overwritten by the one from .zip archive. The destination password will also be overwritten by the one from the .zip archive.

If you do not overwrite the credential, the existing credential remains untouched.
6.3 Importing Configuration Data

To import an archive exported previously:

1. Click **Configuration**, and then click **Import Objects**.

   ![Import Objects dialog box]

2. On the dialog box, click **Browse**, and select a source .zip archive that was previously exported from FlowForce Server.

3. Click **Import**. FlowForce Server displays the records that are about to be imported on a separate page, along with their dependencies. The "Current state" column informs about what will happen to each record after you click the **Import** button.

   ![Import page]

4. Click **Import**.

   If the archive you are importing has external dependencies that cannot be found in the target instance of FlowForce Server, the **Current State** column displays the status "Does not exist". For information about how to address this, see **Handling Missing Dependencies**.
6.4 Handling Missing Dependencies

When you export data from FlowForce Server, you can always exclude certain objects from the export. However, some objects may have dependencies on other objects. If you do not export dependencies together with the object that depends on them, this may lead to errors when you later import that data back into FlowForce Server. The example below is meant to help you understand the implications and how to address them.

Let's assume that you have the two records shown below and choose to export from FlowForce Server only the job, without exporting the credential record:

**Export**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Dependencies</th>
<th>Remote Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>/public/Jobs/AddNumbers</td>
<td>job</td>
<td>Depends on: /public/Jobs/my.credentials</td>
<td>Might or might not exist on remote server</td>
</tr>
<tr>
<td>/public/Jobs/my.credentials</td>
<td>credential</td>
<td>Referenced by: /public/Jobs/AddNumbers</td>
<td>Might or might not exist on remote server</td>
</tr>
</tbody>
</table>

In this case, the following happens:

1. The job will be exported (but it will have a missing dependency)
2. The credential record will not be exported.

If you later attempt to import the .zip archive into a FlowForce Server environment where the object `/public/Jobs/my.credentials` exists, the import is possible because the missing dependency is now resolvable.

**Import**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Dependencies</th>
<th>Current State</th>
</tr>
</thead>
<tbody>
<tr>
<td>/public/Jobs/AddNumbers</td>
<td>job</td>
<td>Depends on: /public/Jobs/my.credentials</td>
<td>Will be created</td>
</tr>
<tr>
<td>/public/Jobs/my.credentials</td>
<td>credential</td>
<td>Referenced by: /public/Jobs/AddNumbers</td>
<td>Exists</td>
</tr>
</tbody>
</table>

However, if the referenced credential does not exist in the target environment, the import dialog box looks as follows:
In the image above, notice that the credential record is no longer recognized as such—namely, its type is "configuration" instead of "credential", and the icon is different. The type "configuration" indicates a generic configuration object (another job or credential), whose type is not known. In this example, the "my.credentials" configuration object was not exported, and the exported package has no information about its exact type*, other than the reference path. Therefore, attempting to import the data above into FlowForce Server will result in an error like: "Operation failed: Path does not exist".

To fix this error, create the missing record at the path indicated by the error message (in this case, the "my.credentials" record), and then perform the import again.

* A credential reference may be a reference to a standalone credential object, and, in some cases, to a job which contains local credentials, see also Referring to Credentials from Jobs.
7 Monitor Job Execution

This topic describes the job monitoring data available on the Home page. When a job meets the trigger criteria or when it is triggered on demand through a Web service call, an instance of that job starts running. The outcome of the job instance, its transition from one status to another, and other execution details are logged by FlowForce Server. You can get information about job execution and its outcome at two levels:

1. From the Home page. The Home page provides a quick overview of the following data related to job execution: Running Jobs, and Active Timers.
2. From the Log View page, which displays all the details about executed jobs.

Running Jobs

The Running Jobs section only displays the first 10 currently running jobs (see screenshot below).

<table>
<thead>
<tr>
<th>Instance ID</th>
<th>Job</th>
<th>Activation Time</th>
<th>Last Action</th>
<th>Status</th>
</tr>
</thead>
</table>

Show all jobs (1), including 1 recently finished, 0 starting and 0 running jobs

The Running Jobs section and its related pages (all jobs, recently finished, starting, and running) contain the following columns:

- **Instance ID**: When a job instance starts, a unique ID is assigned to it. The instance ID helps you track the execution status of each job instance on the Log page. You can click the instance ID inside the table as the job is running—this redirects you to the Log page where you can only view the details of the selected job instance. If you would like to use the job’s instance ID in a job (for example, to create unique file names), this is possible with the help of the instance-id expression function.
- **Job**: This column indicates the path where you can find the configuration of this job instance.
- **Activation Time**: The date and time when the job instance started running.
- **Last Action**: The date and time of the last execution status.
- **Status**: The job status as it was when the page was last refreshed. To find out more about job instance statuses, see Job Statuses.

All jobs

When you click all jobs, a new page called Recent and Running Jobs opens (see screenshot below). The table on this page displays all the running and any recently finished jobs, including jobs that failed. Such jobs are displayed only for a short time (approximately 1-2 minutes) after their execution has finished. You can always check the full history of each job instance from the Log page. For more information, see Log View.
The **Stop job** button is only displayed for running jobs. It enables you to stop running job instances. Note, however, that this may cause data corruption and should be done only exceptionally. For details, see Stop Jobs.

The **Recent and Running Jobs** page is not refreshed automatically. To get the latest status of all jobs, click the **Reload Grid** button.

**Recently finished jobs**
If you click **recently finished**, a new page will open and display only the finished jobs (*see screenshot below*). Jobs remain in this list for 90 seconds.

**Starting jobs**
If you click **starting**, a new page will open and display jobs that are about to start.

**Running jobs**
If you click **running**, a new page will open and display the currently running jobs.

**Active timers**
The **Active Timers** section displays up to 20 of the most recent jobs scheduled to run via **triggers** or as **Web services** (*see screenshot below*).
To view the full list of such jobs, click **Show all active triggers and services**. The table of active triggers and services has the following columns:

- **Type**: Indicates the type of the trigger (*watch*, *timer*, *service*). The *watch* trigger refers to either a file system trigger or an HTTP trigger. See the *Info* column for additional details.
- **Job**: Specifies the path of the job where the trigger or the service is defined. Click the link to open the job's configuration page.
- **Next run**: Applies to watch triggers only. This column indicates when the trigger will run next.
- **Info**: Provides additional information, such as the URL of the Web service, in the case of jobs exposed as Web services. For watch or timer triggers, this column summarizes the current configuration of the trigger.
- **Service URL**: Specifies the URL where the Web service exposed by the respective job is accessible. This only applies to jobs exposed as Web services.

The **Active Triggers and Services** table is not refreshed automatically. Click the button (Reload Grid) to refresh the page.
7.1 Job Statuses

Across its lifetime, a job instance has various statuses, as indicated by the Status column in the Running Jobs table below.

<table>
<thead>
<tr>
<th>Instance ID</th>
<th>Job</th>
<th>Activation Time</th>
<th>Last Action Time</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>/public/MyTask</td>
<td>2019-05-05 12:27:00</td>
<td>2019-05-05 12:27:00</td>
<td>Running step 2, Stop Job</td>
</tr>
</tbody>
</table>

The following diagram illustrates how a job instance typically changes from one state to another across its lifetime. It is assumed that no loss of FlowForce Server service or network interruptions occurred. Note that some of the statuses take a very short time span and will not normally be visible in the user interface.

Job statuses can be divided broadly into two types: *created* and *finished*. Each of these types is further divided into different statuses (see below).

*Created*

*Created* is the first state the job is in before any other action takes place. This status is abstract (i.e. it cannot be entered) and cannot be observed. *Created* is a superset of the following statuses: *Starting, Waiting, Waiting for slot, Running, and Aborting* (see details below).
Monitor Job Execution

**Job Statuses**

- **Starting**: If the execution queue has an opening and the instance is not delayed for some reason, it proceeds to the **Starting** status. The **Starting** status has a short time frame and lasts while the instance starts up. Then the job instance usually switches to the **Running** status.

- **Waiting**: If the instance is delayed, it receives the **Waiting** status.

- **Waiting for slot**: If the job instance is ready to run, but the execution queue is currently full, this job instance switches to the **Waiting for slot** status. An execution queue has a limited number of slots. Therefore, only the specified number of job instances can be executed in parallel in the same queue. Any further instances arriving for that queue will wait until a slot becomes available.

- **Running**: Indicates that the job instance is currently running and will stay in this state until the execution is complete or until some external event occurs that ends the execution prematurely. Except for a very brief time window at the beginning, this status has a step number associated with it. Therefore, the instance gets the **Running step {step}** status. Job instances can also have the following statuses: **Running postponed steps** and **Running postponed step {step}**. To find out more about postponed steps, see this topic.

- **Aborting**: A job instance switches to this status when the user cancels a job. It might take FlowForce Server some time to process the request. The **Aborting** status acknowledges the receipt of this request. Note that the job instance may actually be able to complete successfully before it switches to the **Aborted** state. If this happens, the job will be reported as having finished successfully. If the previous status had a step number, the **Aborting after step {step}** status would be shown instead of **Aborting**.

**Finished**

The **Finished** status is abstract (i.e., it cannot be entered) and includes the following statuses: **Finished successfully**, **Failed**, **Aborted**, **Interrupted**, **Superseded**, **Lost connection**, **Synchronizing**, **Untracked**, **Recovering** (see details below).

- **Finished successfully**: This is a final state which indicates that the job has completed successfully. The status **Finished successfully after step {step}** additionally indicates that the successful completion is associated with a particular step number.

- **Failed**: The execution of the job instance has failed. This is a final status and there will be no further attempts to run the job instance. The **Failed after step {step}** status additionally indicates that the failure is associated with a step number.

- **Aborted**: This status indicates that a user has stopped the job although it may also happen indirectly after an unexpected shutdown. This is a final state that indicates that at least some part of the job has not finished. If the previous status had a step number, the **Aborted after step {step}** status would be shown instead of **Aborted**.

- **Interrupted**: The execution of the job instance has been interrupted. This is a more forceful variation of the **Aborted** state. The job instance cannot be restarted. Therefore, it should be treated as failed. To avoid data inconsistency, it is recommended to check the outcome manually.

- **Superseded**: This status means that the job instance has not executed anything and that some other instance might have run instead of it. This status can only appear before the **Starting** status when, for example, the triggerfile has changed again during the settle time specified by the **Wait N seconds for settle** option. This is an uncritical condition.
- **Lost connection** (*FlowForce Server Advanced Edition only*): This status applies when multiple FlowForce instances run as a cluster. This status indicates that the master machine has lost the connection to the worker machine. When the connection is lost, FlowForce Server does not know whether the instance is still running. When the worker connection is reestablished, the instance switches to the **Synchronizing** status.

- **Synchronizing** (*FlowForce Server Advanced Edition only*): This status applies when multiple FlowForce instances run as a cluster. In a clustered setup, the master machine gets the current progress of job instances from the worker machines. When the worker connection is reestablished, the instance starts synchronizing, and FlowForce is trying to get the latest status from the worker.

- **Untracked**: An instance becomes untracked when FlowForce is terminated while the instance is still running. This status is similar to **Lost connection**, but it is not possible to find out about the state. The **Untracked** status only appears after restarting the service for instances that were still running when FlowForce was terminated.

- **Recovering**: When an instance has become untracked, FlowForce Server will switch on the **Recovering** state before the job instance can proceed.
7.2 Stop Jobs

You can stop any currently running job provided that your user account (or any roles that your user account is member of) have the Stop any job privilege.

**Warning:** Since stopping a job may cause data corruption, it must be done only in exceptional circumstances.

When you stop a job, FlowForce Server first attempts to terminate the job gracefully. While it is not possible to predict the outcome of a stopped job, a graceful termination will normally attempt to perform the error handling tasks associated with the job (if any have been defined). If graceful termination is not possible, FlowForce terminates the job forcefully, after waiting for some time. To force FlowForce Server to terminate the job forcefully earlier, you can click the **Force stop job** button at any time.

**To stop a running job:**

1. Click **Home**. Any currently running jobs are displayed in Running Jobs page section.

2. Click **Stop job**. FlowForce Server prompts you to confirm the action:

   ![Stop running instance of job](image)

3. Click **OK**. Be patient while the system attempts to stop the running instance; this may take several minutes, depending also on the job kind. During this time interval, the job status changes to "Aborting" or "Aborting after step N".
As soon as the job instance is stopped, the status changes to "Aborted", or "Aborted after step N".

If, due to any reason, the job instance could not be stopped gracefully, click **Force stop job** to stop it forcefully.
7.3 Log View

You can view the details logged by FlowForce Server with respect to all kinds of events, including job runs, from the "Log View" page. You can access the "Log View" page either by clicking the Log menu item, or from various other contexts where a View Log button is displayed (for example, each job's configuration page displays a View log button).

Note: By default, you can view the log of any jobs where you have "read" access. To view the global log of all jobs and events in FlowForce Server, your user account must have the View unfiltered log privilege.

Log View

By default, the "Log View" page displays the most recent logged events, with minimum severity set to "Info". You can load older records by clicking the Show 25 more records, Show 100 more records, or Show 500 more records buttons.

The Date column indicates the date and time when the logged event occurred. The Severity column indicates the severity of the logged event, which can be "info", "warning", or "error". The User column specifies the originator of the event (this can be a FlowForce service, the Python security service, or a specific FlowForce user).

Each run of a job produces a unique job instance. The unique job instance is tracked by the Instance ID column. To filter out events by their instance ID, click the link displayed in the Instance ID column. This opens a separate page with all information about the currently selected instance, see Viewing a Log Instance. Note that some logged events do not have a job instance ID because it is not applicable. This is the case, for example, for job configuration change events.
The **Message** column may include links to the configuration page of the job that triggered the event.

To resize any column on the grid, click any of the vertical bars delimiting the column headings, and, holding the left mouse button pressed, drag to the left or right.

The **Copy Permalink to Clipboard** button copies the current URL of the log view to the clipboard, including any selected parameters, for example `?id=2773968&limit=25`. This is useful if you want to quickly load the same information later into the page. For example, you can paste the permanent URL into another browser's address bar, or send it to someone else so that they can see the same log view.

**Exporting logs**

To export the log to a file on the disk, click **Export**. All records that are currently visible in the page will be exported as a zipped JSON (JavaScript Object Notation) file. When you click **Export**, you may either be prompted to save the zip archive to a local directory, or it will be downloaded to your browser's default download directory.

**Filtering logs**

You can filter logs by the following criteria:

- **Date from** - Includes only events after this date.
- **Date to** - Includes only events before this date.
- **Object path** - Includes only events that originate from the job configurations at the selected path. You can select the path to some specific FlowForce object, like a job or credential record.
- **Minimum severity** - Includes events with this severity set as minimum. For example, to include events of all severities, select "Info" as minimum severity. To skip both information messages and warnings, select "Errors" as minimum severity.

If you set both the **Date from** and **Date to** filters, the records within that range are shown, up to 1000 records. To view additional records, click the **Show N more records** buttons. The records having the most recent date are always shown first.

After changing any of the filters, click the **Show** button to apply the filters. Clicking the **Show** button repeatedly loads the most recent logs, while preserving the filters. Clicking any of the **Show N more records** buttons also preserves the currently selected filters.

The **Reset Filter** button clears all filters and refreshes the log. Clicking the **Show** button without any filter selected has the same effect (the log is refreshed).
7.4 Instance Log

The "Instance log" page provides detailed information about a specific logged instance of a job (or "log instance"). You can open this page in one of the following ways:

- Click the instance ID link of a record in the Log view page.
- Click the instance ID link of a record in the Recent and Running section of the home page.
- Click "View Log" from the job configuration page to go to the Log View page, and then click the instance ID link of a record.

Log for instance 4

The "Instance log" page may report the following categories of data, if applicable:

- Messages produced while executing the job instance, grouped by step. This includes:
  - Messages from executing built-in functions or mappings, along with their parameters.
Results of steps that run the `compute` and `compute-string` functions.

Error messages that lead to retry in "Execute with success/failure" handler step or to job failure.

- Information about elapsed time between steps execution.
- Iterations of "for-each" steps
- Information about how many times the job was retried, see `Retry on Error`
- Streams produced by executing mappings or by the `commandline` function.

The Export button creates a .zip archive of all data associated with the current log instance. If you have exported logged instances to a .zip archive on the disk, you can load them for viewing by clicking the View Exported Log for Instance button.

The View Exported Log for Instance button enables you to load data from a previously saved .zip archive into the page.

When you have finished viewing a log instance loaded from a .zip archive, click the Close Exported Log View button.
8 FlowForce Expressions

FlowForce expressions represent custom code that can be computed and executed by FlowForce Server when a job runs. You can think of FlowForce expressions as a basic scripting language understood by FlowForce that helps you "glue together" multiple steps within a job. FlowForce expressions are typically necessary in the following contexts:

- In parameters of built-in functions (that is, you can write or embed expressions in input fields in the job configuration page). Here are just a few examples:
  - Change the data type of the result returned by the execution step
  - Pick a specific value from a result that returns an array of values
  - Concatenate multiple values in order to produce a string.
- In "when" steps, to produce conditional statements. This enables you to execute the step if the expression you provide evaluates to Boolean true.
- In "for-each" steps. "For-each" steps enable you to loop through a sequence of items, where the sequence is defined by an expression.

This section describes the concepts that will help you build FlowForce expressions for scenarios such as the ones listed above.
8.1 Computing an Expression

A simple way to test FlowForce expressions before embedding them in jobs is to create an execution step that calls the `/system/compute` function. For a step-by-step example, see Creating a “Hello, World!” Job.

The `/system/compute` function evaluates the value of the Expression parameter and returns the computed result. Importantly, this function has no defined return type. The actual type depends on the expression being computed. For example, if you pass to this function the expression `1+1`, the function returns the numeric value `2`. However, if you pass to this function the expression `'1+1'`, it returns the string value `'1+1'`.

To understand this concept better, create a step that calls the `/system/compute` function and enter "1+1" in the expression field. Make sure to declare the job return type as "string", as shown below.

![Execution Steps](image)

When you attempt to save the job, FlowForce displays a "Types string and number do not match" error. This error happens because the computed expression is a number, whereas the return type of the job is declared as a string value.

To fix the typing problem, either change the return type of the job to “number” or convert the number to a string. The example below calls the FlowForce expression function `string` which converts a number into a string value.

![Execution Steps](image)

When you need to compute an expression and return the value as string, you can alternatively use the `/system/compute-string` function. In this case, note that the expression part must be delimited from the string with curly braces (see Embedding Expressions in String Fields).
8.2 The Expression Language Rules

To avoid errors in FlowForce expressions, follow these rules:

- Use only allowed or declared values.
- To use a string literally, enclose it within single quotes.
- To embed an expression in a string field, enclose it within curly braces, that is, the { and } characters.
- The expression must produce a data type which is meaningful in the field where the expression was entered.

Let's now have a look at these rules in more detail.

Rule #1: Use only allowed or declared values

The following constructs are allowed in FlowForce expressions:

- FlowForce expression functions (for complete reference, see Expression Functions)
- FlowForce operators (see Operators)
- Numeric values
- String values
- Previously declared variables

When you type text inside a field which allows FlowForce expressions, a real-time syntax check takes place. If the syntax is not correct, FlowForce highlights in red the offending characters. Below is an example of a syntax validation error:

```
Execute function /system/fileSystem/copy

Parameters:
Source: (source)            as string (required) Set to
Target: (target)            as string (required) Set to
```

The error occurs because neither `source` nor `target` have been declared in the job, so FlowForce cannot interpret the expression. The problem can be fixed by declaring these values (for example, as job input parameters):
Rule #2: Enclose strings in single quotes

If you need to use a string literally, enclose it within single quotes. Otherwise, the expression might produce undesired results or validation will fail. Consider the following examples:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Will be evaluated as...</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1+1</td>
<td>2</td>
<td>The data type of the value is numeric.</td>
</tr>
<tr>
<td>'1+1'</td>
<td>1+1</td>
<td>The data type of the value is string.</td>
</tr>
<tr>
<td>1+1==2</td>
<td>true</td>
<td>The data type of the value is Boolean.</td>
</tr>
</tbody>
</table>

When you need to convert values from one data type to another, use the FlowForce expression functions (see also Rule #4).

Rule #3: Use curly braces in string fields

If you want to embed an expression inside a string field, enclose the expression within curly braces. In the example below, curly braces delimit the expression instance-id() (which is a FlowForce expression function) from the rest of the string.

<table>
<thead>
<tr>
<th>Execute function</th>
<th>/system/mail/send</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters: From:</td>
<td>+</td>
</tr>
<tr>
<td>To:</td>
<td><a href="mailto:someone@example.org">someone@example.org</a></td>
</tr>
<tr>
<td>Subject:</td>
<td>Job {instance-id()} has completed.</td>
</tr>
</tbody>
</table>

If the entire field is of type "as expression", do not use curly braces. For example, the Expression parameter of the system/compute built-in function has this type. Below is an example of a correct value for this field (notice no curly braces are used):
Typing curly braces inside the expression field would trigger a syntax error:

See also [Embedding Expressions in String Fields](#).

**Rule #4: Use the correct data type**

Finally, be aware that FlowForce performs data type checks when you save a job. An error will occur if the expression entered in a field does not match the data type expected by the field. You can see the data type expected by each field displayed on the right side of it, for example:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Will be evaluated as...</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</td>
<td>0.25 (as Number)</td>
<td>The data type of the value is numeric. Use this expression in a field or context which expects a numeric value; otherwise, job validation would fail.</td>
</tr>
<tr>
<td>1+1==2</td>
<td>true (as Boolean)</td>
<td>The data type of the value is Boolean. Use this expression in a field or context which expects a Boolean value; otherwise, job validation would fail.</td>
</tr>
<tr>
<td>'apple'</td>
<td>apple (as String)</td>
<td>The data type of the value is string. Use this expression in a field or context which expects a string value; otherwise, job validation would fail.</td>
</tr>
<tr>
<td>concat('1','2','3')</td>
<td>123 (as String)</td>
<td>The data type of the value is string. Use this expression in a field or context which expects a string value; otherwise, job validation would fail.</td>
</tr>
</tbody>
</table>
### Expression Table

<table>
<thead>
<tr>
<th>Expression</th>
<th>Will be evaluated as...</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>1+'apple'</code></td>
<td>-</td>
<td>This expression is not valid, and FlowForce will return an error when you attempt to save the job. Evaluation cannot take place because two different data types (string and numeric) are being compared.</td>
</tr>
</tbody>
</table>
| `{content(\texttt{stdout}(\texttt{result}))}` | `[...] (as String)` | This expression uses two nested expression functions:  
- The function `\texttt{stdout}` gets the standard output of a shell command, as stream.  
- The function `\texttt{content}` converts the stream value to a string.  
Although the expression is correct, the job will validate successfully only when the following is true:  
- The value "result" has been previously declared.  
- The value "result" actually contains the standard output of a shell command.  
- The expression is embedded into a string field.  
See also [Calling Expression Functions](#). |
8.3 Embedding Expressions in String Fields

To use a FlowForce expression in a string field, enclose the expression within curly braces, that is, the "{" and "}" characters. The expression part of a string field normally has a light purple background, which helps you distinguish the expression part from the rest of the string, for example:

```
Execute function /system/mail/send
Parameters: | From: +
| To: someone@example.org
| Subject: Job {instance-id[]} has completed.
```

In a string field, only the expression enclosed within curly braces will be treated by FlowForce as an expression. If you want FlowForce to interpret the "{" and "}" characters literally, write double braces instead of a single brace. Consider the following cases:

<table>
<thead>
<tr>
<th>A string field with the following value...</th>
<th>Will be evaluated as...</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>echo Hello, World!</td>
<td>echo Hello, World!</td>
<td>The string does not use any curly braces (it does not contain an embedded expression), so it is evaluated as is.</td>
</tr>
<tr>
<td>echo {'Hello, World'}</td>
<td>-</td>
<td>The string cannot be evaluated. The embedded expression is not syntactically correct, so FlowForce displays a syntax error.</td>
</tr>
<tr>
<td>echo {'Hello, World!'}</td>
<td>echo Hello, World!</td>
<td>The string contains an embedded expression which is syntactically correct. However, the expression is inside a string field, so the evaluation result would be the same if you used no expression at all (see the first example above).</td>
</tr>
<tr>
<td>echo {{--Hello, World!--}}</td>
<td>echo {'Hello, World!'}</td>
<td>The string does not contain an expression, since the escape characters {{ and }} were used.</td>
</tr>
</tbody>
</table>
8.4 Calling Expression Functions

The FlowForce expression language includes a number of functions that can be used to perform basic operations (primarily, handle values returned by execution steps). You can call these functions from any context where FlowForce expressions are valid (for example, by typing them inside text boxes that represent parameters of a function).

FlowForce expression functions should not be confused with the FlowForce built-in functions. Built-in functions are called from FlowForce execution steps (that is, they are executed as steps), while expression functions are called from FlowForce expressions.

As a typical scenario to call expression functions, let's consider the job illustrated below, which consists of two execution steps.

The first step executes a shell command (namely, it outputs the text "Hello, World"). Notice that the data type returned by this step is "as result". The returned value is declared as `var1`.

The second execution step calls the `/system/compute-string` built-in function. We called this function in order to convert `var1` to a string. The expression itself is embedded into a string field (which is indicated by the curly braces), and it calls two nested expression functions.

- The function `stdout` returns the standard output of a shell command, as stream.
- The function `content` converts the stream value to a string.

Now that the data type conversion is complete, you can further use the string value `var2` as required by your job processing logic (for example, send it in an email).

For reference to all available expression functions, see Expression Functions.
8.5 FlowForce Data Types

FlowForce operates with the following data types.

**string**
Represents a string value, for example: 'Hello, World!'.

**number**
Represents a numeric value, for example: 1, 0, 56, 0.45565.

**Boolean**
Represents a **true** or **false** value.

**result**
This is an abstract type which represents a result produced by an execution step.

An execution step may process various executable files which may be MapForce mappings, StyleVision transformation files, shell functions, and others. The `result` data type, therefore, stands for whatever represents the output of such files.

If the execution step runs a MapForce mapping, the output could be an XML, XBRL, text, JSON, and any other file types generated by MapForce.

If the execution step runs a StyleVision transformation, the output could be PDF, Word, HTML files, and any other output types generated by StyleVision.

To get access to the resulting value, give it some name (for example, "output"), and pass it to the `{results}` expression function. This will convert it to a stream, which you can further process with stream expression functions (see also Calling Expression Functions).

If the execution step runs a shell command, call specific step result expression functions depending on what exactly you need to output. For example, to return the standard output as a stream, use the expression `{stdout(output)}`. To return the standard error as a stream, use the expression `{stderr(output)}`. For more information, see Step Result Functions.

**results**
It may be the case that a MapForce mapping or a StyleVision transformation returns multiple objects. The result produced by such steps has `results` as data type.

To handle such output, use the `{results(output)}` expression function which returns an array of streams. Then pick a particular stream from the array using the `nth` function.

For example, the job illustrated below was created from a StyleVision transformation file deployed to FlowForce. This job takes as input parameter an XML file and returns multiple outputs in various formats.
The job has three execution steps.

1. The first execution step performs the actual data transformation.
2. The second execution step calls the `compute` function of FlowForce to get one of the results of this transformation. Namely, the expression `as-file(nth(results(output), 0))` gets the first item in the array of streams, as a file. The expression uses "0" and not "1" because the array has a zero-based index.
3. The third execution step copies the HTML file to the `c:\archive` directory.

`item`

Sometimes, you need to create expressions that assemble or disassemble lists (see List Functions). A list consists of objects of generic type `item`. An item has an abstract data type. You can determine the data type of `item` depending by looking at the type of objects that make up the list (which can be strings, numbers, or even streams). Note that a list can contain only items of the same data type.

The image below illustrates a loop where "item" is of numeric type, since the list itself consists of numeric values.
For a step-by-step example that utilizes lists, see Copy Files.
### 8.6 Operators

To build FlowForce expressions, you can use the operators listed below. Remember that you can test any expression by calling the built-in function `system/compute`.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>Checks if $a$ and $b$ are equal (numerically equal for numbers, code-point equal for strings).</td>
<td>$2 + 3 == 5$ computes to <code>true</code>&lt;br&gt;$2 + 3 == 4$ computes to <code>false</code></td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Checks if $a$ and $b$ are not equal. Note that the following three expressions are equivalent:</td>
<td>$2 + 2 != 5$ computes to <code>true</code>&lt;br&gt;$3 + 2 != 5$ computes to <code>false</code></td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Checks if $a$ is less than $b$ (numerically less for numbers, see below for strings).</td>
<td>$4 &lt; 5$ computes to <code>true</code></td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Checks if $a$ is less than or equal to $b$.</td>
<td>$5 &lt;= 5$ computes to <code>true</code></td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Checks if $a$ is greater than $b$.</td>
<td>$5 &gt; 1$ computes to <code>true</code></td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Checks if $a$ is greater than or equal to $b$.</td>
<td>$5 &gt;= 5$ computes to <code>true</code></td>
</tr>
<tr>
<td><code>+</code></td>
<td>Addition.</td>
<td>$1 + 1$ computes to 2</td>
</tr>
<tr>
<td><code>-</code></td>
<td>Subtraction.</td>
<td>$2 - 1$ computes to 1</td>
</tr>
<tr>
<td><code>*</code></td>
<td>Multiplication.</td>
<td>$3 * 2$ computes to 6</td>
</tr>
<tr>
<td><code>/</code></td>
<td>Division.</td>
<td>$6 / 3$ computes to 2</td>
</tr>
</tbody>
</table>

String comparisons are performed as follows:

- The common prefix of the two strings are ignored (evaluated on code points)
- If both remaining strings are non-empty, their first code points are compared numerically
- Empty strings are less than non-empty strings

Use parentheses to instruct FlowForce to evaluate the expression inside first. For example:

$2 + 3 * 4$ computes to 14.

$(2 + 3) * 4$ computes to 20.
9 Integration with Altova Product Line

In How It Works, you have seen an overview of Altova products working together. Essentially, mapping files created with Altova MapForce and transformation files created with Altova StyleVision can be automated with the help of the following server counterpart products: MapForce Server (or MapForce Server Advanced Edition) and StyleVision Server. In addition, functions available in RaptorXML Server can also be invoked from FlowForce Server jobs, if the latter runs under FlowForce Server management.

MapForce Server and StyleVision Server can run mappings and transformations across multiple platforms (Windows, macOS, Linux), either at the command line, or from an API call. If these products do not run alongside FlowForce, automation entails developing programs or writing scripts which call the API or invoke the command line of MapForce Server or StyleVision Server.

When MapForce Server and StyleVision Server run under FlowForce Server management, automation can be taken to the next level. Namely, you can deploy the mappings and transformations directly to FlowForce Server and run them as jobs. This way, the mapping or transformation will benefit from all the advantages of a FlowForce Server job: scheduled or on demand execution, execution as a Web service, configuration by means of FlowForce expressions, error handling, conditional processing, email notifications, and so on.

Once deployed to FlowForce Server, the mapping or transformation appears in the container to which you deployed it. As illustrated below, mappings have the .mapping extension while transformations have the .transformation extension.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCalc.transformation</td>
<td>StyleVision transformation</td>
</tr>
<tr>
<td>CompletePO.mapping</td>
<td>MapForce mapping</td>
</tr>
</tbody>
</table>
```

From a FlowForce perspective, such objects are actually functions, and thus can be turned into new jobs. They can also be called from existing jobs, and accept various inputs (typically, files) as parameters. Note that FlowForce Server does not execute such mapping or transformation functions by itself; MapForce Server or StyleVision Server (or both, depending on the case) are invoked to perform the actual execution.

The RaptorXML functions are available in the RaptorXML container, see also Integration with RaptorXML Server.

The next sections discuss how to prepare mappings and transformations for server execution, how to turn them into jobs and how to process their results in FlowForce Server.
9.1 Preparing Files for Server Execution

A mapping designed and previewed with MapForce may refer to resources which are outside of the current machine and operating system (such as databases). In addition to this, in MapForce, all mapping paths follow Windows-style conventions by default. Thirdly, the machine where MapForce Server runs might not support the same database connections as the machine where the mapping was designed. For this reason, running mappings in a server environment typically requires some preparation, especially if the target machine is not the same as the source machine.

Note: The term "source machine" refers to the computer where the MapForce is installed and the term "target machine" refers to the computer where MapForce Server or FlowForce Server is installed. In the most simple scenario, this is the same computer. In a more advanced scenario, MapForce runs on a Windows machine whereas MapForce Server or FlowForce Server runs on a Linux or macOS machine.

As best practice, always make sure that the mapping validates successfully in MapForce before deploying it to FlowForce Server or compiling it to a MapForce Server execution file.

If MapForce Server runs standalone (without FlowForce Server), the required licenses are as follows:

- On the source machine, MapForce Enterprise or Professional edition is required to design the mapping and compile it to a server execution file (.mfx).
- On the target machine, MapForce Server or MapForce Server Advanced Edition is required to run the mapping.

If MapForce Server runs under FlowForce Server management, the following requirements apply:

- On the source machine, MapForce Enterprise or Professional edition is required to design the mapping and deploy it to a target machine.
- Both MapForce Server and FlowForce Server must be licensed on the target machine. The role of MapForce Server is to run the mapping; the role of FlowForce is to make the mapping available as a job which benefits from features such as scheduled or on demand execution, execution as a Web service, error handling, conditional processing, email notifications, and others.
- FlowForce Server must be up and running at the configured network address and port. Namely, the "FlowForce Web Server" service must be started and configured to accept connections from HTTP clients (or HTTPS if configured) and must not be blocked by the firewall. The "FlowForce Server" service must also be started and running at the designated address and port.
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user).

General considerations

- If you intend to run the mapping on a target machine with standalone MapForce Server, all input files referenced by the mapping must be copied to the target machine as well. If MapForce Server runs under FlowForce Server management, there is no need to copy files manually. In this case, the instance and schema files are included in the package deployed to the target machine.
- If the mapping includes database components which require specific database drivers, such drivers must be installed on the target machine as well. For example, if your mapping reads data from a Microsoft Access database, then Microsoft Access or Microsoft Access Runtime (https://www.microsoft.com/en-us/download/details.aspx?id=50040) must be installed on the target machine as well.
Preparing Files for Server Execution

- When you deploy a mapping to non-Windows platforms, ADO, ADO.NET and ODBC database connections are automatically changed to JDBC. Native SQLite and native PostgreSQL connections are preserved as such and require no additional configuration. See also "Database connections" below.
- If the mapping contains custom function calls (for example, to .dll or .class files), such dependencies are not deployed together with the mapping, since they are not known before runtime. In this case, copy them manually to the target machine. The path of the .dll or .class file on the server must be the same as in the "Manage Libraries" window in MapForce, for example:

![Manage Libraries](image)

- Some mappings read multiple input files using a wildcard path. In this case, the input file names are not known before runtime and so they are not deployed. For the mapping to execute successfully, the input files must exist on the target machine.
- If the mapping output path includes directories, those directories must exist on the target machine. Otherwise, an error will be generated when you execute the mapping. This behavior is unlike MapForce, where non-existing directories are generated automatically if the option Generate output to temporary files is enabled.
- If the mapping calls a Web service that requires HTTPS authentication with a client certificate, the certificate must be transferred to the target machine as well.
- If the mapping connects to file-based databases such as Microsoft Access and SQLite, the database file must be manually transferred to the target machine or saved to a shared directory which is accessible to both the source and the target machine and referenced from there, see "File-based databases" below.

Making paths portable

If you intend to run the mapping on a server, ensure that the mapping follows the applicable path conventions and uses a supported database connection.

To make paths portable to non-Windows operating systems, use relative instead of absolute paths when designing the mapping in MapForce:

1. Open the desired mapping design file (.mfd) with MapForce on Windows.
2. On the File menu, select Mapping Settings, and clear the Make paths absolute in generated code check box if it is selected.
3. For each mapping component, open the Properties dialog box (by double-clicking the component's title bar, for example), and change all file paths from absolute to relative. Also, select the Save all file paths relative to MFD file check box. For convenience, you can copy all input files and schemas in the same folder as the mapping itself, and reference them just by the file name.
For more information about dealing with relative and absolute paths while designing mappings, refer to MapForce documentation.

Importantly, both MapForce Server and FlowForce Server support a so-called "working directory" against which all relative paths will be resolved. The working directory is specified at mapping runtime, as follows:

- In FlowForce Server, by editing the "Working-directory" parameter of any job.
- In MapForce Server API, through the `WorkingDirectory` property of the COM and .NET API, or through the `setWorkingDirectory` method of the Java API.
- In MapForce Server command line, the working directory is the current directory of the command shell.

**Database connections**

Be aware that ADO, ADO.NET, and ODBC connections are not supported on Linux and macOS machines. Therefore, if the target machine is Linux or macOS, such connections are converted to JDBC when you deploy the mapping to FlowForce or when you compile the mapping to a MapForce Server execution file. In this case, you have the following options before deploying the mapping or compiling it to a server execution file:

- In MapForce, create a JDBC connection to the database
- In MapForce, fill the JDBC database connection details in the "JDBC-specific Settings" section of the database component.

If the mapping uses a native connection to a PostgreSQL or SQLite database, the native connection is preserved and no JDBC conversion takes place. If the mapping connects to a file-based database, such as Microsoft Access and SQLite, additional configuration is required, see "File-based databases" below.

Running mappings with JDBC connections requires that the Java Runtime Environment or Java Development Kit be installed on the server machine. This may be either Oracle JDK or an open source build such as Oracle OpenJDK.

- The `JAVA_HOME` environment variable must point to the JDK installation directory.
- On Windows, a Java Virtual Machine path found in the Windows registry will take priority over the `JAVA_HOME` variable.
- The JDK platform (64-bit, 32-bit) must be the same as that of MapForce Server. Otherwise, you may get an error with the reason: "JVM is inaccessible".

**To set up a JDBC connection on Linux or macOS:**

1. Download the JDBC driver supplied by the database vendor and install it on the operating system. Make sure to select the 32-bit version if your operating system runs on 32-bit, and the 64-bit version if your operating system runs on 64-bit.
2. Set the environment variables to the location where the JDBC driver is installed. Typically, you will need to set the `CLASSPATH` variable, and possibly a few others. To find out which specific environment variables must be configured, check the documentation supplied with the JDBC driver.

**Note:** On macOS, the system expects any installed JDBC libraries to be in the `/Library/Java/Extensions` directory. Therefore, it is recommended that you unpack the JDBC driver to this location; otherwise, you will need to configure the system to look for the JDBC library at the path where you installed the JDBC driver.
Oracle Instant Client connections on macOS

These instructions are applicable if you connect to an Oracle database through the Oracle Database Instant Client, on macOS. Prerequisites:

- Java 8.0 or later must be installed. If the Mac machine runs a Java version prior to Java 8, you can also connect through the JDBC Thin for All Platforms library, and disregard the instructions below.
- Oracle Instant Client must be installed. You can download the Oracle Instant Client from the Oracle official download page. Note that there are several Instant Client packages available on the Oracle download page. Make sure to select a package with Oracle Call Interface (OCI) support, (for example, Instant Client Basic). Also, make sure to select the 32-bit version if your operating system runs on 32-bit, and the 64-bit version if your operating system runs on 64-bit.

Once you have downloaded and unpacked the Oracle Instant Client, edit the property list (.plist) file shipped with the installer so that the following environment variables point to the location of the corresponding driver paths, for example:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASSPATH</td>
<td>/opt/oracle/instantclient_11_2/ojdbc6.jar:/opt/oracle/instantclient_11_2/ojdbc5.jar</td>
</tr>
<tr>
<td>TNS_ADMIN</td>
<td>/opt/oracle/NETWORK_ADMIN</td>
</tr>
<tr>
<td>ORACLE_HOME</td>
<td>/opt/oracle/instantclient_11_2</td>
</tr>
<tr>
<td>DYLD_LIBRARY_PATH</td>
<td>/opt/oracle/instantclient_11_2</td>
</tr>
<tr>
<td>PATH</td>
<td>$PATH:/opt/oracle/instantclient_11_2</td>
</tr>
</tbody>
</table>

Note: Edit the sample values above to fit the paths where Oracle Instant Client files are installed on your operating system.

File-based databases

File-based databases such as Microsoft Access and SQLite are not included in the package deployed to FlowForce Server or in the compiled MapForce Server execution file. Therefore, if the source and target machine are not the same, take the following steps:

1. In MapForce, right-click the mapping and clear the check box Make paths absolute in generated code.
2. Right-click the database component on the mapping and add a connection to the database file using a relative path. A simple way to avoid path-related issues is to save the mapping design (.mfd file) in the same directory as the database file and to refer to the latter from the mapping just by file name (thus using a relative path).
3. Copy the database file to a directory on the target machine (let's call it "working directory"). Keep this directory in mind since it will be required to run the mapping on the server, as shown below.

To run such mappings on the server, do one of the following:

- If the mapping will be run by MapForce Server under FlowForce Server control, configure the FlowForce Server job to point to the working directory created previously. The database file must reside in the
working directory. For an example, see Exposing a Job as a Web Service.

- If the mapping will be run by standalone MapForce Server at the command line, change the current directory to the working directory (for example, `cd path\to\working\directory`) before calling the `run` command of MapForce Server.

- If the mapping will be run by the MapForce Server API, set the working directory programmatically before running the mapping. To facilitate this, the property `WorkingDirectory` is available for the MapForce Server object in the COM and .NET API. In the Java API, the method `setWorkingDirectory` is available.

If both the source and the target machines are Windows machines running on the local network, an alternative approach is to configure the mapping to read the database file from a common shared directory, as follows:

1. Store the database file in a common shared directory which is accessible by both the source and the target machine.
2. Right-click the database component on the mapping and add a connection to the database file using an absolute path.

**Global Resources**

If a mapping includes references to Global Resources instead of direct paths or database connections, you will be able to use Global Resources on the server side as well. When you compile a mapping to a MapForce Server execution file (.mfx), the references to Global Resources will be kept intact, so that you can provide these on the server side, at mapping runtime. When deploying a mapping to FlowForce Server, you can optionally choose whether it should use resources on the server.

For mappings (or mapping functions, in case of FlowForce Server) to run successfully, the actual file, folder, or database connection details that you supply as Global Resources must be compatible with the server environment. For example, files and folders paths must use the Linux convention for paths if the mapping will run on a Linux server. Likewise, Global Resources defined as database connections must be possible on the server machine.

For further information, see Resources.

**XBRL Taxonomy Packages**

When you deploy a mapping that references XBRL Taxonomy Packages to FlowForce Server, MapForce collects all external references from the mapping and then resolves them using the current configuration and currently installed taxonomy packages. If there are resolved external references that point to a taxonomy package, then the taxonomy package is deployed together with the mapping. FlowForce Server will use that package—as it was during deployment—to execute the mapping. To refresh the taxonomy package used by FlowForce Server, you will need to change it in MapForce and redeploy the mapping.

Note that the root catalog of MapForce Server influences the way taxonomies are resolved on the target machine. The root catalog is found at the following path relative to the MapForce Server installation directory: `etc/RootCatalog.xml`.

Taxonomy packages that were deployed with a mapping will be used if the root catalog of MapForce Server does not already contain such package or does not contain a package that is defined for the same URL prefix. The root catalog of MapForce Server has priority over the deployed taxonomy.

If MapForce Server runs standalone (without FlowForce Server), it is possible to specify the root catalog that should be used by the mapping as follows:
At the command line, this is possible by adding the option `-catalog` to the `run` command.

In the MapForce Server API, call the method `SetOption`, and supply the string "catalog" as first argument, and the path to the root catalog as second argument.

If a mapping uses XBRL components with table linkbases, the taxonomy package or the taxonomy package configuration file must be supplied to the mapping at runtime, as follows:

- At the MapForce Server command line, add the option `--taxonomy-package` or `--taxonomy-packages-config-file` to the `run` command.
- In the MapForce Server API, call the method `SetOption`. The first argument must be either "taxonomy-package" or "taxonomy-packages-config-file". The second argument must be the actual path to the taxonomy package (or taxonomy package configuration) file.
9.2 Deploying Mappings to FlowForce Server

Deploying a mapping to FlowForce Server means that MapForce organizes the resources used by the specific mapping into an object and passes it through HTTP (or HTTPS if configured) to the machine where FlowForce Server runs. MapForce mappings are typically deployed to FlowForce Server in order to automate their execution by means of FlowForce Server jobs. Once a mapping is deployed, you can create a full-featured FlowForce Server job from it, and benefit from all job-specific functionality (for example, define custom triggering conditions for the job, expose it as a Web service, and so on).

Note: The term "source machine" refers to the computer where the MapForce is installed and the term "target machine" refers to the computer where FlowForce Server is installed. In the most simple scenario, this is the same computer. In a more advanced scenario, MapForce runs on a Windows machine whereas FlowForce Server runs on a Linux or macOS machine.

The package deployed to FlowForce includes the following:

- The mapping itself. After deployment, the mapping becomes available in the FlowForce Server administration interface as a mapping function (.mapping), at the path you specify. Any source components become input arguments, and any target components become output arguments of this function.

- All kinds of input instance files (XML, CSV, Text) that are used by the mapping.

Prerequisites
See Preparing Mappings for Server Execution.

Deploying the mapping to FlowForce Server

1. Run MapForce and ensure that the transformation language is set to BUILT-IN (either click the Built-in toolbar button or select the Output | Built-in Execution Engine menu command).
3. Enter your deployment settings (as described below), and click OK. If you selected the **Open web browser to create new job** check box, the FlowForce Server administration interface opens in the browser, and you can start creating a FlowForce Server job immediately.

The following table lists the mapping deployment settings available on the Deploy Mapping dialog box.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Server, Port, Use SSL**     | Enter the server host name (or IP address) and port of FlowForce Server. These could be **localhost** and **8082** if FlowForce Server is running on the same machine at the default port. When in doubt, log on to FlowForce Server Web administration interface and check the I.P. address and port displayed in the Web browser's address bar.  
  If you encounter connectivity errors, ensure that the machine on which FlowForce Server runs is configured to allow incoming connections on the designated address and port.  
  To deploy the mapping through a SSL-encrypted connection, select the **Use SSL** check box. This assumes that FlowForce Server is already                                                                 |
Setting | Description
--- | ---
 | configured to accept SSL connections. For more information, refer to FlowForce Server documentation ([https://www.altova.com/documentation](https://www.altova.com/documentation)).
User and Password | The user name and password to be entered depends on the value of the Login drop-down list (see next option). If the Login drop-down list is set to `<Default>` or `Directly`, enter your FlowForce Server user name and password. Otherwise, enter your domain user name and password, and select the domain name from the Login drop-down list.
Login | If Directory Service integration is enabled in FlowForce Server, select the domain name from this drop-down list, and enter your domain credentials in the User and Password fields (see previous option).
Use Resources, Resource Path | Select the Use Resources check box if the mapping function should use Resources after it is deployed to the server. If you select the check box, you must also enter the path of the respective resource on the server in the Resource Path text box. To select the resource, click the Ellipsis button.
 | If there are no resources on the server yet to choose from, click Deploy Global Resources, and deploy the required Global Resource to server.
 | If you do not select the Use Resources check box, any Global Resources will be resolved, based on the currently selected configuration. On the server, the mapping function will no longer require Global Resources, but will use the resolved value instead.
Path | Click Browse, and select the path where the mapping function should be saved in FlowForce Server container hierarchy. By default, the path is set to the `/public` container of FlowForce Server.
 | From the dialog box, you can also create new containers or delete existing containers and mappings, provided that you have the required FlowForce Server permissions and privileges.
Save mapping before deploying | This option is available if you are deploying an unsaved mapping. Select this check box to save the mapping before deployment.
Open browser to create new job | If you select this check box, the FlowForce Server Web administration interface opens in the browser after deployment, and you can start creating a FlowForce Server job immediately.

**Troubleshooting**
The following table lists problems that you might encounter when deploying a mapping, and their solution.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploying the mapping returns the following error:</td>
<td>Make sure that, on the target machine, the FlowForce Web Server service is running and configured to listen</td>
</tr>
</tbody>
</table>
Problem | Solution
---|---
I/O operation on file ... failed. I/O Error 28: Failed to connect to <server> port 8082. Timed out System error 10060: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond. | for connections on the specified port (8082, by default). Also, make sure that the firewall does not block incoming connections through this port. The FlowForce Server service must be running as well in order for the deployment to be possible.

Deploying the mapping returns the following error: I/O operation on file ... failed. I/O Error 413: Payload Too Large | This error may occur if an input file of the deployed mapping exceeds the maximum size limit of HTTP requests allowed by FlowForce Server (roughly 100 MB). You can increase the limit by setting the max_request_body_size option (in bytes) in the flowforceweb.ini and flowforce.ini files. For details, see Configuration File Reference.

Selecting the server version (Windows only)
If the server where you deploy the mapping has multiple versions of MapForce Server running under FlowForce Server management (applicable to Windows servers only), then you are additionally prompted to specify the version of MapForce Server with which you want this mapping to be executed.

Note: The dialog box appears when the FlowForce Server installation directory contains .tool files for each MapForce Server version which runs under FlowForce Server management. By default, a MapForce Server .tool file is added automatically to this directory when you install MapForce Server as part of FlowForce Server installation. The path where the .tool files are stored in FlowForce is: C:\Program Files\Altova\FlowForceServer2022\tools. If you have additional versions of MapForce Server which you want to run under FlowForce Server management, their .tool files may need to be copied manually to the directory above. The .tool file of MapForce Server can be found at: C:\Program Files\Altova\MapForceServer2022\etc.
9.3 Running Mappings and Transformations as Jobs

You can create a FlowForce Server job from a MapForce mapping or StyleVision transformation as follows:

1. First, deploy the mapping or transformation to FlowForce Server. This step is done in MapForce (and StyleVision, respectively):
   - On the **File** menu, click **Deploy to FlowForce (Server)**.
   
   For reference to the deployment settings, see Deploying Mappings to FlowForce Server.

2. In FlowForce Server, navigate to the FlowForce container where you deployed the mapping or transformation (for example, the container “/public”).

3. Click the required mapping or transformation, and then click **Create Job**. Alternatively, you can refer to the mapping or transformation from an existing job, by entering its path in the **Execute function** box:

You can now configure the job according to your needs. For example, you can run it as a **Web service** or with the help of a **trigger**. For a step-by-step example which illustrates deploying a StyleVision transformation and creating a job from it, see Creating a Job from a StyleVision Transformation. For a similar example for MapForce, see Creating a Job from a MapForce Mapping. For an example job which calls both MapForce Server and StyleVision Server, see Example: Generating Multiple PDFs from Multiple XMLs.

One of the most important parts of running a transformation or mapping job is handling the job input files. There are two approaches you can take: supply the input files statically to the job, or supply them dynamically at job runtime (for example, from a path). The exact approach to use depends on your needs. If your job needs to run...
with the same input data every time, then the first approach is suitable. Otherwise, if you need your FlowForce jobs to pick up data from files supplied dynamically from a path, then the second approach must be used.

**MapForce mappings**

In case of mappings deployed from MapForce, any instance files (such as XML, CSV, JSON, Excel, and so on) are deployed together with the mapping and implicitly packaged as static. This means that, when the job runs, FlowForce will read data from the statically packaged files by default, which might not always be what you need. There are two scenarios here:

1. If you right-click the mapping in MapForce and select the **Make paths absolute in generated code** check box before deploying the mapping, all the input files explicitly appear with the prefix `altova://packagefile/` in FlowForce Server.

   To instruct FlowForce Server not to read data from packaged files, remove the prefix `altova://packagefile` from the path. You can then refer to the file using either an absolute or a relative path. If using a relative path, the path is relative to the **Working Directory** parameter. For example, if you intend to provide as input some files from `C:\FlowForce\CompletePO`, then set the working directory to `C:\FlowForce\CompletePO` and enter just the name of the input files, as shown below.

2. If the **Make paths absolute in generated code** check box is NOT selected before deploying the mapping to FlowForce, the input files are shown with their relative path in FlowForce. Note that FlowForce will still read data from the packaged file in this case as well, even when there are files with the same name in the working directory. To instruct FlowForce not to read data from the packaged file, you can either make the file paths absolute or supply them as parameters to the job, as shown below:
Alternatively, you can change the mapping design in MapForce so that the input file names are input parameters to the mapping. For example, the mapping illustrated below takes both the input and output file names as parameters.

When deployed to FlowForce Server, the parameters appear as such in the job configuration page (the files themselves are not packaged).
The mapping illustrated above is called **FileNamesAsParameters.mfd** and is one of the example files that ship with MapForce. For information about how this mapping is designed, refer to the MapForce documentation.

**StyleVision transformations**

In case of StyleVision transformations, you can handle input files as follows:

1. Open the PXF (Portable XML Form) file in StyleVision. If you have a SPS (StyleVision Power Stylesheet), StyleVision will prompt you to convert it to PXF format when you attempt to deploy it to FlowForce Server.
2. In the Design Overview window, click **Configure embedded files**. A dialog box appears.

3. Notice the option **Embed the working XML file**. If you select this check box, the working XML file will be part of the deployed package and, by default, FlowForce Server will read data from it each time when the job runs. A packaged file is indicated as such in FlowForce:

To supply the file dynamically to the job, remove the prefix `altova://packagedfile/` or change the path to an absolute one. If using a relative path, the path is relative to the **Working Directory** parameter. Alternatively, clear the **Embed the working XML file** check box before deploying the transformation to FlowForce Server.

If you clear the **Embed...** check box for resources like CSS files or images, FlowForce Server will look for them in the job working directory.
9.3.1 Credentials in Mapping Functions

Earlier in this documentation, you have seen an introduction to Credentials. Recall that it is possible to create credentials not only in FlowForce Server, but also at mapping design time, in MapForce.

When you deploy a mapping containing credentials from MapForce to FlowForce Server, the credentials are deployed to the server as well. The deployed information will contain only the fields that you filled in when creating the credential record. For example, this may be an empty credential (if you chose to store only the credential name) or a credential object that contains both the username and password.

You can also deploy credential objects from MapForce to FlowForce Server as standalone objects, separately from the main mapping. You can choose directly from MapForce the target container where they should be deployed. For more information, refer to MapForce documentation (https://www.altova.com/documentation).

The field Password is considered sensitive data.

The sensitive data will be deployed only if you selected the Include in MapForce Server Execution File and Mapping Deployment check box at mapping design time in MapForce. This applies both when you deploy the mapping and when you deploy the standalone credentials.

In FlowForce Server, you can see whether a mapping function needs credentials by opening the page of the respective mapping function, for example:

![Function Input Parameters]

If you selected the Include in MapForce Server Execution File and Mapping Deployment check box when creating the credential, then the job will use the credentials deployed together with the mapping. In this case, you don’t need to specify them from the job configuration page. For example, the following execution step will run the mapping function with the stored credentials if such exist (notice that the "my.credentials" parameter is not expanded):
You can always override the stored credentials with any other credential object that was defined directly in FlowForce Server, or with some local credentials. To do this, click the "+" button and either select a credential object that already exists in FlowForce Server, or enter the username and password directly, for example:

The credentials supplied as parameter to the execution step take precedence over credentials stored inside the mapping function.

If you did not select the **Include in MapForce Server Execution File and Mapping Deployment** check box when creating the credential in MapForce, it is mandatory to supply credentials as parameters to the execution step; otherwise, the job execution will fail.

### 9.3.2 Dynamic Authentication

In MapForce, it is possible to configure mappings that call Web services for basic HTTP authentication. Dynamic authentication is one of the ways to achieve this; it is an alternative to using credentials. Dynamic authentication means designing the mapping so that it accepts the username and password as input parameters. For details about configuring dynamic authentication, refer to MapForce documentation ([https://www.altova.com/documentation](https://www.altova.com/documentation)).

When you deploy a mapping containing dynamic authentication to FlowForce Server, the username and password become input parameters to the mapping function. Any FlowForce Server job that calls such a mapping function will require the username and password before it can run successfully, for example:
In the example illustrated above, the username and password are simply entered in the respective text boxes. However, you can also supply them as input parameters to the job, see Managing Input Parameters.

### 9.3.3 Resources

Altova Global Resources are portable references to files, folders, or databases. When stored as Global Resources, paths and database connection details become reusable and available across multiple Altova applications. For example, if you frequently need to open the same file in multiple Altova desktop applications, you may find it convenient to define it as a Global Resource. This way, you don't even need to remember the file path because you can select the respective Global Resource from the "Open File" dialog box instead. This also has the advantage that, if the file path ever changes, you will change it in one place only.

A typical usage of Global Resources is to define a database connection once and reuse it across all Altova applications that support Global Resources. For example, you can create a database connection on the machine where a MapForce mapping was designed and then reuse the same connection on the machine where MapForce Server runs the mapping (this may require, in some cases, that both machines have the same database client software installed).

Optionally, you can create multiple variations of the same Global Resource (known as "configurations"). This lets you easily switch file or folder paths (or even databases) depending on your needs. For example, you could create a "database" resource with two configurations: "development" and "production". Taking FlowForce Server as example, you could then easily retrieve data from either the development or production database, by supplying the desired configuration to the mapping function.

You can create Global Resources from the following Altova desktop applications: Altova Authentic, DatabaseSpy, MobileTogether Designer, MapForce, StyleVision, and XMLSpy. On the server side, Global Resources can be consumed by the following Altova server applications: FlowForce Server, MapForce Server, RaptorXML Server, RaptorXML+XBRL Server.

For more information about creating Global Resources, refer to the "Altova Global Resources" chapter of MapForce documentation.

### Resources in FlowForce Server

From the FlowForce Server perspective, Global Resources cease to be "global" to some extent, because they are not stored in one XML file as in desktop applications. In FlowForce, each resource is a reusable object that
may contain file or folder paths or database connection details. Resources may be copied, exported, and imported, and are subject to the same user access mechanism like other FlowForce Server objects. In other words, any FlowForce user could use any resource in their mapping functions, if they have the required permissions.

Once you have created a mapping with Global Resources in MapForce, you can deploy it to FlowForce Server. At deployment time, you can choose whether the mapping should use Global Resources, by selecting the **Use Resources** check box on the deployment dialog box. If you do not select the check box, any Global Resources used by the mapping will be resolved, based on the currently selected configuration. If you selected the check box, the mapping function will require resources in FlowForce Server as well. The following is an example of a mapping function deployed to FlowForce that requires resources to run (notice that the first parameter gets the default file path from a resource):

As illustrated above, in FlowForce Server, it is the mapping function that consumes Global Resources, not the job. The exact meaning of the verb “consumes” depends on how exactly the mapping was designed with respect to Global Resources. For example, the mapping may have been configured to read from or write to a path, in case of files or folders. In the example above, the mapping function reads the path of the first input file from the resource. This means that all jobs using this function will use the same path, unless you override the path from the job configuration page.

You can also deploy Global Resources to FlowForce Server as standalone objects. In other words, there is no need to deploy a mapping first in order to be able to deploy a Global Resource. For more information about deploying Global Resources to FlowForce Server, refer to MapForce documentation [here](https://www.altova.com/documentation).

**Structure of resources**

In all Altova desktop applications, global resources are maintained as XML files. The default file is called *GlobalResources.xml*; you can find it in the `C:\Users\<username>\Documents\Altova` directory on the computer where MapForce is installed. A Global Resource file may contain multiple resources, also known as "aliases". An alias is either a file path, or a directory path, or a group of database connection details. Aliases, in their turn, can have multiple configurations. As described previously, configurations enable you to switch
paths or databases. This is best understood by looking at the structure of the following sample Global Resource file (note some data was omitted for simplicity):

```xml
<Resources>
  <Resource Alias="MyFile">
    <Configurations>
      <Configuration Location="C:\test.json" ContentKind="File" Configuration="Default"/>
      <Configuration Location="C:\production.json" ContentKind="File" Configuration="Production"/>
    </Configurations>
  </Resource>
  <Resource Alias="MyDirectory">
    <Configurations>
      <Configuration Location="C:\Test" ContentKind="Folder" Configuration="Default"/>
      <Configuration Location="C:\Production" ContentKind="Folder" Configuration="Production"/>
    </Configurations>
  </Resource>
  <Resource Alias="MyDatabase">
    <Configurations>
      <Configuration ContentKind="DataSource" Configuration="Default">
        <DatabaseContextInfo vendor="sqlite" connection="C:\Resources\Test.sqlite"/>
      </Configuration>
      <Configuration ContentKind="DataSource" Configuration="Production">
        <DatabaseContextInfo vendor="sqlite" connection="C:\Resources\Production.sqlite"/>
      </Configuration>
    </Configurations>
  </Resource>
</Resources>
```

The file above defines three resources (aliases): a file path called "MyFile", a directory path called "MyDirectory", and a SQLite database called "MyDatabase". Each alias has two configurations: a default configuration used for testing, and a production configuration.

In FlowForce Server, because of the specifics of the multi-user server environment, resources work slightly differently. Specifically, an XML resource file such as the one above becomes a resource object in FlowForce. Inside the resource object, there can be multiple aliases, just like in desktop applications. However, each alias has only one configuration, and that is the configuration that you've selected upon deploying the resource from MapForce to FlowForce Server.

Whenever you deploy Global Resources from MapForce to FlowForce Server, only one of the configurations is deployed at a time.

For example, if you deployed the global resource file above, either the "Default" or "Production" configuration will be deployed to the server (not both at the same time). If you choose the "Default" configuration, the resource object would look as follows in FlowForce Server:
In FlowForce, any mapping function can consume one specific configuration of a global resource. Therefore, in this example, if you need the "Production" configuration on the server, you should deploy the same resource file once again, this time selecting the configuration "Production" from the deployment dialog box in MapForce. Alternatively, you can create a resource directly on the server, as described below, and change the mapping function to point to it instead of the "Default" resource. Note, however, that the alternative approach is possible with file and directory resources, not with databases.

**Changing the resource of a mapping function**

In FlowForce, resource objects are identified by the icon. Therefore, if you've deployed both the "Default" and the "Production" configurations from the example above, the corresponding resources in FlowForce Server may appear as follows:
To change the resource used by a mapping function:

1. Go to the container where the mapping function was deployed and click to open the function.
2. Under “Resources”, select a new resource path. Selecting resources works in the same way as with other FlowForce objects such as functions, credentials, and so on.

If the mapping function does not have a “Resources” section, this mapping was not configured for Global Resources in MapForce (or the Use Resources check box was not selected on deployment).

Any mapping function can use any resource, if the following requirements are satisfied:

- The resource kind is compatible with the function. For example, a “folder” resource is not suitable if the mapping function needs a “file” resource.
- The resource alias name is the one required by the mapping function. You normally select the alias name at mapping design time, in MapForce, but you can also override it in FlowForce, as further described below.

Resources and job configuration

As stated before, resources are consumed at mapping function level, not at job configuration level. When a job runs, it consumes those global resources that are defined in the function called by the job. Therefore, when you edit a job from the job configuration page, you have only very minimal configuration options with respect to resources, like “Overriding the resource alias” (further described below).

In some cases, it may be possible to reference a resource (like a folder or file) directly from the job configuration page. Please note that this may not work in all contexts and should be generally avoided unless you have a very good reason to use such references.

**Note:** It is not supported to refer to a resource from the “Working Directory” parameter of an execution step. This is because processing of resources requires that the MapForce Server process be already started, whereas the working directory is set before MapForce Server starts.

Overriding the resource alias

Even if a file or folder resource can have multiple aliases, only one of them is used at job runtime. The alias used at runtime is the one selected in MapForce while designing the mapping. For example, the following MapForce component is configured to generate **output.csv** to a directory alias called "MyDirectory". If you deploy this mapping to FlowForce Server, the mapping function on the server must also point to a resource that contains the "MyDirectory" alias.
As an alternative to editing the mapping in MapForce whenever you need to change the alias, you can also override the alias in FlowForce Server, from the job configuration page. To override file or folder aliases in a job, use the following syntax, replacing `MyFile` or `MyDirectory` with the required alias name:

<table>
<thead>
<tr>
<th>Resource kind</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td><code>altova://file_resource/MyFile</code></td>
</tr>
<tr>
<td>Directory</td>
<td><code>altova://folder_resource/MyDirectory</code></td>
</tr>
</tbody>
</table>

For example, in the job configuration below, the directory alias was changed to "TestDir".

**Note:** Overriding the alias as shown above is not supported for database resources. If you have multiple databases aliases, switch to the required database alias in MapForce before deploying the mapping to FlowForce Server.

**Creating resources**

You can create only file or folder resources in FlowForce Server. To create a global resource in FlowForce Server, open a container of choice and click **Create | Create Resource**.
Creating database resources is not supported in a server environment. To create database resources, use the Global Resources editor of MapForce or any other Altova desktop application that supports Global Resources, and then deploy the resources from MapForce to FlowForce Server.

The resource alias should match the one required by the mapping function where you will use this resource. Otherwise, you will need to tweak jobs manually so that they point to the correct alias, as described above in “Overriding the resource alias”.

Within the same resource object, you can create multiple aliases if required, by clicking the New File Resource or New Folder Resource buttons. This is optional, however. If you create multiple aliases, remember that you will need to modify jobs so as to indicate which alias it should use.

Editing resources
You can edit file or folder resources directly in FlowForce Server, as an alternative to doing this in MapForce and deploying them again. To edit a resource, click the respective record, update the paths (or the database connection details), and then click Save.

Note: In case of database resources, you can edit in FlowForce only certain fields such as the connection string or default database. It is, however, not possible to change the database vendor and connection method.
Updating a resource affects with immediate effect all of the following:

- All the mapping functions referencing that resource
- All the jobs that call the respective mapping functions.
9.4 Accessing the Mapping/Transformation Result

After a MapForce mapping or StyleVision transformation has been deployed to FlowForce Server, it becomes a FlowForce function which can be called from other execution steps. For example, in the first step of the job below, a mapping function called \texttt{SimpleTotal.mapping} is being executed.

Notice that the job consists of two steps:

1. Step 1 calls MapForce Server to actually run the \texttt{SimpleTotal.mapping} function. Importantly, the \texttt{Assign this step's result to} field gives a name to the mapping result (in this case, it is \texttt{output}; however, it can be any name you choose).
2. Step 2 calls the /system/compute function which converts the output of the mapping to a stream.

By default, the output of a mapping or transformation function is of generic type \texttt{result}. In order for the output to become useful, \texttt{result} must be converted to whatever data type you require (for example, string, stream, file). For this purpose, the /system/compute built-in function is available, as well as various FlowForce expression functions. In the example above, the built-in function /system/compute was called to perform the required data type conversion. Namely, the expression \texttt{stdout(output)} converts the result of the previous step to a stream.

The table below lists examples of FlowForce expressions that you will likely need to process the result of a mapping or a transformation function. Remember that, in all these examples, \texttt{output} is the name you entered in the \texttt{Assign this step's result to} field.

<table>
<thead>
<tr>
<th>FlowForce Expression</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdout(output)</td>
<td>Converts \texttt{output} to a stream.</td>
</tr>
<tr>
<td>content(stdout(output))</td>
<td>Converts \texttt{output} to string.</td>
</tr>
</tbody>
</table>
### FlowForce Expression

<table>
<thead>
<tr>
<th>Expression</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>as-file(stdout(output))</code></td>
<td>Converts <code>output</code> to a file.</td>
</tr>
<tr>
<td><code>as-file(nth(results(output), 0))</code></td>
<td>This kind of expression is required if <code>output</code> consists of multiple files. This happens when the mapping or transformation function was designed (in MapForce or StyleVision) to generate not just a single output, but multiple outputs. The expression converts <code>output</code> to a sequence of streams, picks up the first stream from the sequence, and converts it to file. For an example, see <a href="450">Creating a Job from a StyleVision Transformation</a>.</td>
</tr>
<tr>
<td><code>as-file(nth(results(output, &quot;CompletePO&quot;), 0))</code></td>
<td>Same as above, except that the file is retrieved from the sequence of streams not by its zero-based index as above, but by name (in this case, &quot;CompletePO&quot;).</td>
</tr>
</tbody>
</table>

For complete reference to FlowForce expression functions that are available to handle the result of a step or job, see [Step Result Functions](450). For an introduction to FlowForce expressions, see [FlowForce Expressions](438).
9.5 Integration with RaptorXML Server

When RaptorXML is integrated into FlowForce, all the functions exposed by RaptorXML Server become available to FlowForce so that you can call them in jobs. More specifically, the RaptorXML functions exist in the /RaptorXML container of FlowForce. In case of RaptorXML+XBRL Server, the container name is /RaptorXMLXBRL.

You can call the RaptorXML functions from jobs similar to calling FlowForce built-in functions:

- In the /RaptorXML (or /RaptorXMLXBRL) container, open the function of interest, and then click Create Job. You can either reference generic functions such as /RaptorXML/valjson or release-specific functions such as /RaptorXML/2022/valjson. The differences between the two are described below.
- Create a new execution step in a job, and call the desired RaptorXML function from an execution step. For example, the step below calls the valjson function:

For examples of jobs that call RaptorXML Server, see:

- Validate a Document with RaptorXML
- Validate XML with Error Logging
- Use RaptorXML to Pass Key/Value Parameter Pairs
For reference to all the RaptorXML functions, refer to the RaptorXML Server documentation (https://www.altova.com/documentation).

Manual integration
Integration between FlowForce Server and RaptorXML Server takes place automatically in many cases (for example, when you run the FlowForce Server installation on Windows and choose to install RaptorXML Server as well). However, there are also cases when manual integration between the two is necessary. Manual integration is typically required when FlowForce Server and RaptorXML Server of different versions were installed separately. For example, if the function definitions of a specific RaptorXML Server version are missing from the FlowForce Server interface even though that version of RaptorXML Server is installed, then manual integration is required.

To perform a manual integration, run the script available at the following path: `{RaptorXML installation directory}\etc\functions\integrate.bat`.

**Note:** On Unix systems, the script name is `integrate.cs`. Superuser privileges (sudo) are required to run this script.

This script takes two arguments: the path to the FlowForce Server installation directory and the path to the FlowForce Server data directory (see FlowForce Server Application Data). When you run the script, the following happens:

- All the release-specific functions of the integrated RaptorXML Server version become available to FlowForce Server so you can call them as jobs.
- The generic (release-agnostic) RaptorXML functions are updated to point to the release-specific functions of the integrated RaptorXML version.

If the script returns errors, the function definitions of the integrated RaptorXML version are not compatible with FlowForce Server. In the unlikely event that this happens, please contact support.

Generic versus release-specific RaptorXML functions
The functions available in the RaptorXML or RaptorXMLXBRL containers are organized as follows:

- Functions from the /RaptorXML container are backward compatible down to the 2014 version of FlowForce Server (which is the first version supporting RaptorXML functions). These generic functions act as wrappers to the release-specific functions from the /RaptorXML/{Release} container. They are guaranteed to be compatible between releases but they do not provide all the features of the latest installed RaptorXML Server.
- Functions from the /RaptorXML/{Release} containers provide all the features of the corresponding RaptorXML release. These functions are compatible with FlowForce Server of the same release. However, any version of RaptorXML Server is not necessarily compatible with any version of FlowForce Server. You can check compatibility by running an integration script (as described under "Manual integration").

If a job calls a generic RaptorXML function, the function acts as a wrapper to the equivalent release-specific function of the RaptorXML Server. The selected RaptorXML release is the one that was most recently integrated into FlowForce, including manually-integrated releases. Still, as mentioned above, such calls will not benefit from the latest RaptorXML features (such as new arguments or even functions). To make use of the latest RaptorXML features from FlowForce jobs, call a release-specific function directly.
A release-specific function determines which RaptorXML .tool file should be used in order to look up the RaptorXML executable. A separate .tool file exists for each RaptorXML Server release. A .tool file instructs FlowForce Server about the location of the RaptorXML Server executable and can also be used to set environment variables, see Setting Environment Variables.

If your FlowForce jobs refer to version-specific RaptorXML functions, and if you would like to upgrade to a newer version of FlowForce Server and RaptorXML Server, you can either modify all the jobs to point to the latest release-specific RaptorXML functions, or you can map the Raptor.tool file to a newer version of the RaptorXML Server executable, as follows:

1. Copy the `Raptor_<release>.tool` file from `{installation}/etc` directory of RaptorXML Server of the latest installed release to the `{configuration data}/tools` directory of FlowForce Server of the same release.
2. Rename the file to match the version of the old release (the Raptor release your jobs are pointing to). For example, if the old release is RaptorXML 2017r3, then rename the file to `Raptor_2017r3.tool`.

If you take the mapping approach, all the existing jobs will continue to look as if they call RaptorXML 2017r3 functions, whereas the .tool file will map in fact to the latest RaptorXML Server executable.
9.6 Tool Files

When you install other Altova servers alongside FlowForce Server, for example, by selecting the relevant server products in the FlowForce Server installation wizard or installing these server products using their stand-alone installer later, a .tool file is installed for each application that runs under FlowForce Server management. The following Altova products can run under FlowForce Server management: MapForce Server, StyleVision Server and RaptorXML Server. Usually, you do not need to configure .tool files unless you need to change environment variables such as CLASSPATH for MapForce Server and StyleVision Server.

FlowForce Server uses .tool files to locate and configure the execution of the other server applications under its management. FlowForce Server searches for .tool files in the application data directory, referred to as DATADIR, and the installation directory, referred to as INSTALLDIR. FlowForce Server first scans DATADIR and then INSTALLDIR. The tables below show the paths of these directories for different operating systems.

<table>
<thead>
<tr>
<th>FlowForce Server application data directory (DATADIR)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux /var/opt/Altova/FlowForceServer2022/data</td>
<td></td>
</tr>
<tr>
<td>macOS /var/Altova/FlowForceServer2022/data</td>
<td></td>
</tr>
<tr>
<td>Windows C:\ProgramData\Altova\FlowForceServer2022\data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FlowForce Server installation directory (INSTALLDIR)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux /opt/Altova/FlowForceServer2022/</td>
<td></td>
</tr>
<tr>
<td>macOS /usr/local/Altova/FlowForceServer2022/</td>
<td></td>
</tr>
<tr>
<td>Windows C:\Program Files\Altova\FlowForceServer2022\ C:\Program Files (x86)\Altova\FlowForceServer2022\</td>
<td></td>
</tr>
</tbody>
</table>

DATADIR is usually an empty directory, where you place any customized tool files. The INSTALLDIR directory is managed by the installation process, and the .tool files contained in it must not be edited.

Information messages

FlowForce Server groups running tool process instances and manages them as configured in the .tool files. When FlowForce enforces the rules regarding the lifetime of tool process instances, all these events may produce information messages in the log. For example:

Starting instance (id) of {tool} for {session}.
Starting {commandline}.
Instance (id) of {tool} for {session} is now idle.
Shutting down instance (id) of {tool} for {session}; sitting idle for too long.
Shutting down instance (id) of {tool} for {session}; maximum reuse count reached.
Instance (id) of {tool} for {session} unexpectedly ceased communication.
Instance (id) of {tool} for {session} attached to job instance {instanceid}.

The information messages listed above do not indicate licensing or queueing issues. Instead, they make it possible to track down potential problems, for example, by offering information about processes that were running at a particular time. If steps or jobs fail, this will generate a separate log message.
Tool file editing
Files with a .tool extension can be edited in a text editor (e.g., Notepad++). The following editing options are available:

1. The executable path under the [Tool] section. Changing this path might be necessary in certain cases, for example, when you need to make .tool files of older versions execute newer versions, or vice versa.
2. The [Environment] section. You can add or edit this section in order to define environment variables required by the tool. For more information, see the subsection below.

Important:
- When you edit a .tool file in DATADIR, changes take effect at once. You do not have to restart FlowForce Server.
- Do not change any .tool file settings other than the ones mentioned above, unless advised by Altova Support.
- It is not possible to define custom tools.

Environment variables
When MapForce Server mappings or StyleVision Server stylesheets run under FlowForce Server management, they may require setting environment variables. For example, you need to set CLASSPATH to specify the location of the JDBC drivers when connecting to a database. To set environment variables required by MapForce Server mappings or StyleVision Server transformations, edit the .tool file of the respective Altova server product. To edit the .tool file, first check if it already exists in the DATADIR directory. If the .tool file does not exist in DATADIR, copy it from INSTALLDIR of FlowForce Server.

You would find .tool files in the INSTALLDIR directory only if MapForce Server or StyleVision Server were installed after FlowForce Server. If the .tool file exists neither in DATADIR nor in INSTALLDIR, it is likely that FlowForce Server was installed after MapForce Server or StyleVision Server. In this case, you can find the .tool file in the etc directory relative to the MapForce Server or StyleVision Server installation directory.

You can add the required environment variables under the [Environment] section in the .tool file. The environment variables set in the .tool file override the environment variables defined by other means. The example of a .tool file (Linux) which sets the CLASSPATH variable is given below:

```
[Environment]
CLASSPATH=/usr/local/jdbc/oracle/ojdbc6.jar
```

Note: If you run the migratedb command while upgrading to a new major version of FlowForce, any .tool files from the application data directory of the previous version will be copied over to the application directory of the new version. This may have unwanted consequences. Therefore, make sure that the application data directory contains the .tool files that you actually need.

For information about executing shell commands or scripts as FlowForce Server jobs, see the /system/shell/commandline function.
10 Command Line Interface

As mentioned previously, the FlowForce solution consists of two services: FlowForce Server and FlowForce Web Server. For each of these services, an executable is available that supports administrative commands that can be run at the command line. You can find both executables at the following path:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>/opt/Altova/FlowForceServer2022/bin/</td>
</tr>
<tr>
<td>macOS</td>
<td>/usr/local/Altova/FlowForceServer2022/bin/</td>
</tr>
<tr>
<td>Windows</td>
<td>&lt;ProgramFilesFolder&gt;\Altova\FlowForceServer\bin\</td>
</tr>
</tbody>
</table>

The executable names are as follows:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Executable Names</th>
</tr>
</thead>
</table>
| Linux    | flowforceserver  
            flowforcewebserver |
| macOS    | flowforceserver  
            flowforcewebserver |
| Windows  | FlowForceServer.exe 
            FlowForceWebServer.exe |

The command line interface (CLI) can be used for administration purposes (such as licensing, troubleshooting, and internal database backup). The commands supported by the CLI are listed below.

Note: The abbreviations "FFS" and "FFW" indicate whether the command is applicable to the FlowForce Server or FlowForce Web Server executable, respectively.

<table>
<thead>
<tr>
<th>Command</th>
<th>FFS</th>
<th>FFW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td>Yes</td>
<td>Yes</td>
<td>Displays help for the command supplied as argument.</td>
</tr>
<tr>
<td>assignlicense</td>
<td>Yes</td>
<td></td>
<td>This command is applicable to Windows platforms only. It can be used to upload and assign a license file to FlowForce Server.</td>
</tr>
<tr>
<td>compactdb</td>
<td>Yes</td>
<td></td>
<td>Reduces the size of FlowForce .db files if they contain deleted records.</td>
</tr>
<tr>
<td>createdb</td>
<td>Yes</td>
<td></td>
<td>Creates a new FlowForce database.</td>
</tr>
<tr>
<td>debug</td>
<td>Yes</td>
<td>Yes</td>
<td>Starts the application in debug mode.</td>
</tr>
<tr>
<td>exportresourcestrings</td>
<td>Yes</td>
<td>Yes</td>
<td>Exports all application resource strings to an XML file</td>
</tr>
<tr>
<td>foreground</td>
<td>Yes</td>
<td>Yes</td>
<td>Starts the application in foreground mode.</td>
</tr>
<tr>
<td>initdb</td>
<td>Yes</td>
<td></td>
<td>Creates or updates the FlowForce database.</td>
</tr>
<tr>
<td>install</td>
<td>Yes</td>
<td>Yes</td>
<td>Installs the application as a Windows service.</td>
</tr>
<tr>
<td>Command</td>
<td>Usage</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>licenseserver</td>
<td>Yes</td>
<td>Registers FlowForce Server with the Altova LicenseServer on the local network.</td>
<td></td>
</tr>
<tr>
<td>migratedb</td>
<td>Yes</td>
<td>Migrates FlowForce Server data from a previous version to the latest version.</td>
<td></td>
</tr>
<tr>
<td>repair</td>
<td>Yes</td>
<td>Starts the application in repair mode.</td>
<td></td>
</tr>
<tr>
<td>resetpassword</td>
<td>Yes</td>
<td>Resets the password of the root user to the default value, and grants to the root user all privileges.</td>
<td></td>
</tr>
<tr>
<td>setdeflang</td>
<td>sd</td>
<td>Yes</td>
<td>Sets the default language.</td>
</tr>
<tr>
<td>start</td>
<td>Yes</td>
<td>Starts the application as a service.</td>
<td></td>
</tr>
<tr>
<td>uninstall</td>
<td>Yes</td>
<td>Uninstalls the application as a Windows service.</td>
<td></td>
</tr>
<tr>
<td>upgradedb</td>
<td>Yes</td>
<td>Upgrades the FlowForce Server database to the latest version.</td>
<td></td>
</tr>
<tr>
<td>verifylicense</td>
<td>Yes</td>
<td>This command is applicable to Windows platforms only. It can be used to verify whether FlowForce Server is licensed, and, optionally, whether a given license key is already assigned to FlowForce Server.</td>
<td></td>
</tr>
</tbody>
</table>

**Conventions**

By convention, this documentation omits the full path of the executable when describing a given command, and uses `flowforceserver` instead of the executable name, for example:

```
flowforceserver help
```

Where `flowforceserver` is the path or name of the executable. Note that, if you use an absolute path, you will be able to run commands regardless of the current directory that your command prompt window (terminal) is in. However, if you would like to call the executable just by typing its name, make sure to do one of the following first:

- Change the terminal's current directory to the FlowForce Server installation directory
- Add the directory where the executable is to the PATH environment variable.

Both of these scenarios are described in more detail below.

**Tips and tricks**

If you are new to command line, be aware of the following tips and tricks.

- To find out the current directory where you command line window is, enter `pwd` on Linux and macOS. On Windows, enter `echo %CD%`.
- Make use of the Tab key to quickly enter various file or directory paths without having to type them in full. For example, if you type `cd c:\prog` at the command line, and then press Tab, you will get `c:\Program Files` automatically pre-filled (or perhaps some other directory under C:\ whose name begins with "Prog").
When entering paths that contain white space, such as C:\Program Files on Windows, enclose them within quotes.

- If you see a message similar to "This command is not recognized as an internal or external command, operable program or batch file", you have most likely mistyped a path or command.
- On Linux, make sure that you use the correct case for file or directory names. For example, typing a path such as /home/nikita/downloads will return an error if the directory name is actually /home/nikita/Downloads.
- When typing a path on Linux or macOS, use forward slashes, as opposed to back slashes on Windows.

### How to run a command

1. Open a command prompt window.
   
   a. To open a command prompt on Windows, press the Windows key and then start typing cmd. Click the Command Prompt suggestion that appears.
   
   b. To open a terminal on Mac, click the Finder icon, and then select Go > Utilities from the menu. Double-click the Terminal icon in the Utilities window.
   
   c. If you run Linux from a graphical user interface, locate and run the Terminal command as applicable to your Linux distribution. If you run Linux from a command line interface, ignore this step.

2. Enter the full path to the executable, followed by the command you want to run. For example, the command below provides help at the command line.

   - Linux: /opt/Altova/FlowForceServer2022/bin/flowforceserver help
   - macOS: /usr/local/Altova/FlowForceServer2022/bin/flowforceserver help
   - Windows: C:\Program Files (x86)\Altova\FlowForceServer2022\bin\FlowForceServer.exe help

   In the example above, the command help was run without any options or arguments. Other commands may have arguments and options, and those arguments and options could be mandatory or optional. Check the reference section for details about each command.

### Calling FlowForce Server in the installation directory

To call the executable without having to type the full path, change the current directory to the directory where the FlowForce Server executable was installed, for example:

   - Linux: cd /opt/Altova/FlowForceServer2022/bin
     .flowforceserver help
   
   - macOS: cd /usr/local/Altova/FlowForceServer2022/bin
     .flowforceserver help
   
   - Windows: cd "C:\Program Files (x86)\Altova\FlowForceServer2022\bin\FlowForceServer.exe"
     .flowforceserver help

You can now run any command by typing just the executable name, for example:

   - Linux: ./flowforceserver help
   
   - macOS: ./flowforceserver help
| Windows | FlowForceServer.exe help |

**Note:** On Linux and macOS systems, the prefix `/` indicates that the executable is in the current directory.

**Calling FlowForce Server from any directory**

To call the executable from any directory, refer to it using the absolute path. Alternatively, if you want to call the program by typing just the executable name, you can edit the PATH environment variable of your operating system so that it includes the full path to the FlowForce Server installation directory. For ways to change the PATH environment variable, refer to the documentation of your operating system.

**Note:** After changing the PATH environment variable, you may need to close the terminal window and open a new one, in order for the changes to take effect.
10.1 help

Purpose
Provides help information about the command supplied as argument.

Syntax

```
FlowForceServer help Command
```

Note: On Linux systems, use an all-lowercase flowforceserver to call the executable.

Arguments
The help command takes a single argument: the name of the command for which help is required. It displays the correct syntax of the command and other information relevant to the correct execution of the command.

Example

```
FlowForceServer help createdb
```

Using --help as option for other commands
Help information about a command is also available by using the --help option with that command. For example, using the --help option with the createdb command, as follows:

```
FlowForceServer createdb --help
```

has the same result as:

```
FlowForceServer help createdb
```
10.2 assignlicense

Purpose
This command is applicable to Windows platforms only. It can be used to upload and assign a license file to FlowForce Server.

Syntax

```
FlowForceServer assignlicense [options] FILE
```

Arguments

| FILE | Specifies the path of the license file to be uploaded. |

Options

| --t, --test-only=true|false | When set to true, the license is uploaded and validated. |
| | | When set to false, the license is uploaded, validated, and assigned as well. |
| | | If this option is not specified, the default value is true. |
10.3 compactdb

Purpose
Reduces the size of FlowForce .db files if they contain deleted records. This command is particularly useful after running the archive-log or truncate-log system maintenance functions.

Syntax

FlowForceServer compactdb [options]

Note: On Linux systems, use an all-lowercase flowforceserver to call the executable.

Options

|--datadir=VALUE

VALUE is the path of the data directory which contains the .db files to be compacted. If this option is not specified, the /data directory will be used by default (see also FlowForce Server Application Data).
10.4 createdb

Purpose
Creates a new database. If the database already exists then the command will fail. The default database is created at installation time, so it is usually not necessary to use this command.

Syntax

FlowForceServer createdb [options]

Note: On Linux systems, use an all-lowercase flowforceserver to call the executable.

Options

|--datadir=VALUE  VALUE is the path of the data directory.
10.5 debug

Purpose
Not for general use. This command starts FlowForce Server in debug mode (that is, not as a service). To stop this mode, press CTRL+C.

Syntax

```
FlowForceServer debug [options]
```

**Note:** On Linux systems, use an all-lowercase `flowforceserver` to call the executable.

Options

```
--datadir=VALUE
```

**VALUE** is the path of the data directory.
10.6 exportresourcestrings

Purpose
Outputs an XML file containing the resource strings of FlowForce Server. It takes two arguments: (i) the language of the resource strings in the output XML file, and (ii) the path and name of the output XML file. Valid export languages (with their language codes in parentheses) are: English (en), German (de), Spanish (es), and Japanese (ja).

Syntax

```
FlowForceServer exportresourcestrings Language XMLOutput
```

**Note:** On Linux systems, use an all-lowercase flowforceserver to call the executable.

Arguments

<table>
<thead>
<tr>
<th>Language</th>
<th>Specifies the language of resource strings in the exported XML file. Allowed languages are: en, de, es, ja</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLOutput</td>
<td>Specifies the location and name of the exported XML file.</td>
</tr>
</tbody>
</table>

Example

This command creates a file called Strings.xml at c:\ that contains all the resource strings of the FlowForce Server application in English.

```
FlowForceServer exportresourcestrings en c:\Strings.xml
```
10.7 foreground

Purpose
Not for general use. This command starts Altova FlowForce Server in the foreground. It is used internally by the startup scripts for Linux.
10.8 initdb

Purpose
Create a new database, or updates an existing one to the latest version. The default database is created at installation time, so it is usually not necessary to use this command.

Syntax

```bash
FlowForceServer initdb [options]
```

**Note:** On Linux systems, use an all-lowercase `flowforceserver` to call the executable.

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--datadir=VALUE</code></td>
<td>VALUE is the path of the database directory.</td>
</tr>
</tbody>
</table>
10.9 install

Purpose
This command is executed by the FlowForce Server installer automatically and it is not available for general use. The command installs Altova FlowForce Server as a service, on Windows. This command does not apply to Linux and macOS.
10.10 licenseserver

Purpose
Registers FlowForceServer with LicenseServer. You must have Administrator privileges (root) to register FlowForce Server with LicenseServer. For more information, see the LicenseServer documentation.

Syntax

```
FlowForceServer licenseserver [options] Server-Or-IP-Address
```

Note: On Linux systems, use an all-lowercase flowforceserver to call the executable.

Example

```
FlowForceServer licenseserver DOC.altova.com
```

The command above specifies that the machine named `DOC.altova.com` is the machine running Altova LicenseServer. If LicenseServer is running on the user's machine, the following commands would also be valid:

```
FlowForceServer licenseserver localhost
FlowForceServer licenseserver 127.0.0.1
```

Options

The options are listed below, in their short forms (first column) and long forms (second column), together with their descriptions. On the command line, one or two dashes can be used for both short and long forms.

```
--j   --json
Prints the result of the registration attempt as a machine-parseable JSON object.
Form: --json=true|false
```
10.11  migratedb

Purpose
Copies FlowForce Server data from a previous application data directory to the current one, and also upgrades the FlowForce database to the latest version if necessary. This command is invoked by the FlowForce installation scripts when there is already a previous version of FlowForce Server installed, so you don't typically need to run it. Running this command may be useful when migrating FlowForce Server to a new machine, or when restoring the application data directory from a backup, see Backup and Recovery.

If you only need to upgrade the FlowForce database version to the latest one, it is sufficient to run upgradedb.

Syntax

FlowForceServer migratedb [options]

Note: On Linux systems, use an all-lowercase flowforceserver to call the executable.

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--datadir=VALUE</td>
<td>VALUE is the path of the database directory</td>
</tr>
<tr>
<td>--olddatadir=VALUE</td>
<td>VALUE is the path of the database directory</td>
</tr>
</tbody>
</table>

Example
To migrate data from the application data directory of FlowForce Server 2020 to FlowForce Server 2022, run:

"C:\Program Files (x86)\Altova\FlowForceServer2022\bin\FlowForceServer.exe" migratedb
--datadir=C:\ProgramData\Altova\FlowForceServer2022\data --olddatadir=C:\ProgramData\Altova\FlowForceServer2020\data
10.12 repair

Purpose
Starts FlowForce Server with all triggers and job execution processes disabled, to enable troubleshooting.

Syntax

```
FlowForceServer repair [options]
```

**Note:** On Linux systems, use an all-lowercase `flowforceserver` to call the executable.

Options

```
--datadir=VALUE
```

**VALUE** is the path of the database directory.

Example

```
FlowForceServer repair --datadir=C:\ProgramData\Altova\FlowForceServer2022\data
```
10.13 resetpassword

Purpose
Resets the password of the \texttt{root} user to the default value, and grants to the \texttt{root} user all privileges. It is recommended to stop the running instance of FlowForce Server before performing this operation (see instructions for starting or stopping services on Linux, macOS, and Windows).

Syntax

```
FlowForceServer resetpassword [options]
```

\textbf{Note:} On Linux systems, use an all-lowercase flowforceserver to call the executable.

Options

```
--datadir=VALUE
```

\textit{VALUE} is the path of the database directory.

Example

```
FlowForceServer resetpassword --datadir=C:\ProgramData\Altova\FlowForceServer2022\data
```
### 10.14 setdeflang (sdl)

**Purpose**
The `setdeflang` command (short form is `sdl`) sets the default language of FlowForce Server. To change the default language, run this command for both `FlowForceServer` and `FlowForceWebServer` services (see **Syntax**).

**Syntax**

```
FlowForceServer setdeflang | sdl LanguageCode
FlowForceWebServer setdeflang | sdl LanguageCode
```

**Note:** On Linux systems, use an all-lowercase `flowforceserver` to call the executable.

The possible values of `LanguageCode` are as follows.

- `en` English
- `es` Spanish
- `de` German
- `fr` French
- `ja` Japanese

**Example**

```
FlowForceServer setdeflang de
```
10.15 start

Purpose
Starts FlowForce Server as a service. This command is used internally by the startup scripts or by the Windows service installation; it is not for general use.

Syntax
FlowForceServer start [options]

Note: On Linux systems, use an all-lowercase flowforceserver to call the executable.

Options
--datadir=VALUE  VALUE is the path of the database directory.
10.16  uninstall

Purpose
This command is executed by the FlowForce Server installer automatically and it is not available for general use. The command uninstalls Altova FlowForce Server as a service, on Windows. The command does not apply to Linux and macOS.
10.17 upgradedb

Purpose
Upgrades the database to the latest version. The default database is upgraded automatically at installation time, so it is usually not necessary to run this command manually.

Syntax

```
FlowForceServer upgradedb [options]
```

Note: On Linux systems, use an all-lowercase `flowforceserver` to call the executable.

Options

```
--datadir=VALUE
```

VALUE is the path of the database directory.

Example

```
FlowForceServer upgradedb --datadir=C:\ProgramData\Altova\FlowForceServer2022\data
```
10.18 verifylicense

Purpose
This command is applicable to Windows platforms only. It can be used to verify whether FlowForce Server is licensed, and, optionally, whether a given license key is already assigned to FlowForce Server.

Syntax

```
FlowForceServer verifylicense [options]
```

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--l, --license-key=VALUE</td>
<td>This option enables you to verify if a particular license key is already assigned to FlowForce Server. The value must be set to the license key that you wish to verify.</td>
</tr>
</tbody>
</table>
11  Built-in Functions

This chapter provides reference information about the system functions built into FlowForce Server. The built-in functions allow you to copy or move files, create directories, execute shell commands, and perform various other actions. The FlowForce Server built-in functions are available in the /system container.

The following topics describe the built-in functions in groups, according to their path relative to the root container.

- /system/abort
- /system/compute
- /system/compute-string
- /system/filesystem
- /system/ftp
- /system/mail
- /system/maintenance
- /system/shell

If RaptorXML Server is integrated into FlowForce Server, then an additional container with all RaptorXML Server functions is available. The same applies for RaptorXML+XBRL Server. For more information, see Integration with RaptorXML Server.

Referring to Windows network paths

When you create jobs, you will typically need to refer to file paths on the machine where FlowForce Server runs or to file paths on the network. When you refer to a Windows network path (e.g., a mapped network drive), use the Universal Naming Convention (UNC) syntax. This is necessary because drive letters are not global to the system, and each logon session is assigned its own drive letters.

The UNC has the following syntax: \\server\sharedfolder\filepath, where server references the server name in the network (defined by the DNS); sharedfolder references a label defined by the administrator (e.g., admin$ is generally the root directory of the operating system installation); filepath refers to the subdirectories below the share.
11.1 /system

The /system container includes all the FlowForce built-in functions. Only the `abort`, `compute`, `compute-string` and `create-file` functions are found directly in this container. Other functions are located in sub-containers according to their area of applicability (for example, AS2 functions, file system functions, mail functions, and so on).

11.1.1 abort

Full path: /system/abort

Aborts the execution of a job. This function is typically used inside a condition (that is, a Choose step) to deliberately end the job when that condition is true. It is the equivalent of a throw or raise function in a programming language. This function does not return a value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>string</td>
<td>Mandatory string parameter. Specifies the message to output when aborting the job.</td>
</tr>
</tbody>
</table>

Examples

In the following job, the `abort` function is used to finish the job with an error if the value of a checked list exceeds 10 items. If the number of items in the list is less than or equal to 10, the job writes the text “The list has less than 10 items” to a file on the local system.
11.1.2 `compute`

Full path: `/system/compute`
Computes the result of an expression and returns the computed value. The computed value can be used in parameters or expressions of other execution steps. You can also use this function to define the output of a job that is used as a service (see the example).

This function returns the value `T0`, which indicates an arbitrary type. That is, the returned data type will be inferred from the expression used in the `Expression` parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Expression of T0</td>
<td>The FlowForce Server expression to be computed. For more information about expressions, see <a href="#">The FlowForce Expression Language</a>.</td>
</tr>
</tbody>
</table>

**Examples**

This example illustrates a job with two execution steps. The first step runs a shell command in the `c:\temp` directory, and the result is declared as `hello`.

Next, this result is passed to the second execution step. The second execution step uses expression language (in particular, the `stdout` and `content` functions) to do the following:

- get the standard output of the result of the first step
- convert the output to string

The `compute` function evaluates the expression entered in the `Expression` text box.

**Execution Steps**

Execute function `/system/shell/commandline`

Parameters:
- Command: `echo "hello world"`
- Working directory: `c:\temp`

Assign this step's result to `hello`

Execute function `/system/compute`

Parameters:
- Expression: `content(stdout(hello))`

Assign this step's result to `name`

See also [Creating a “Hello, World” Job](#).
11.1.3 compute-string

Full path: /system/compute-string

Outputs the result of an expression as a string. This step function does essentially the same as the compute function, except that the input format is a string instead of an expression.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>string</td>
<td>The FlowForce Server expression (as string) to be computed.</td>
</tr>
</tbody>
</table>

Examples

To understand the difference between the /system/compute/ and /system/compute-string functions, consider the following example:

In the job illustrated above, there are three execution steps.

The first step calls the /system/compute/ function. Notice that no curly braces were used. The entire field stores an expression (as suggested by the background color), so curly braces are implied. The expression
concatenates two values and produces a string depending on the job input parameter. For example, if the input parameter is "c:\temp\invoices.txt", the step will return the string value "invoices.txt" (declared as outputname1).

The second step calls the /system/compute-string function. This function processes a string which contains an embedded FlowForce expression. Here, curly braces are used to delimit the expression from the rest of the string. Notice that the embedded expression has a background color other than the rest of the string. Although a different technique was used, the step result (outputname2) is the same as outputname1.

Finally, the third step calls the /system/compute-string function again, in order to compare the outputname1 with outputname2. If both values are identical, the result will be the string value "Both expression are identical". Otherwise, the result will be "Both expressions are not identical".

11.1.4 create-file

The create-file function allows you to store stream content in a file that you may need to use in the future. Files created with the help of the create-file function are not temporary. Such files belong to the user and not to FlowForce.

The create-file function is similar to the as-file function in that it creates the specified target file with the specified stream content, but create-file does not create any temporary files. Use /system/create-file to store stream content that you intend to keep. Use as-file to pass the stream content as a file to some program. This might be a temporary file managed by FlowForce.

The source expression (see screenshot below) can be anything that returns a stream. You can use anything you can pass to the as-file function. For example, you could use the following options:

- stdout(result) , stderr(result) , result(result, name, index) get streams out of step results;
- mime-flatten(stream) produces a message/rfc822 stream from another by prefixing it with its MIME headers;
- mime-multipart(string, stream*) produces a MIME multi-part structure as a stream;
- stream-open(filename, contenttype) opens a file on disk;
- empty-stream() produces a zero-length stream;
- stream-from-string(text, encoding, content-type) encodes a string value into a stream.

Example

The screenshot below illustrates the create-file function. Our goal is to create a file called CreateTest.txt and save it on the desktop. We are going to use the stream-from-string function, which encodes a string value into a stream. As a result, we will see our new CreateTest.txt file containing the string MyFileContent.

Note: To run the job, set a trigger and/or run the job as a service.
Execution Steps

- Execute function: `/system/create-file`
- Parameters:
  - Target file: `CreateTest.txt`
  - Source stream: `stream-from-string('MyFileContent')`
  - Overwrite target: **on**
  - Working directory: `Desktop`
  - Abort on error: **on**
11.2  /system/filesystem

The /system/filesystem container includes functions used to manage files and directories on the operating system where FlowForce Server runs.

All file paths in job execution steps must be paths on the operating system where FlowForce Server runs, not on your local machine.

11.2.1  copy

Full path: /system/filesystem/copy

Copies a file from a source to a target directory. Optionally, the file can be copied with a new name to the target directory. When invoked from a simple execution step, this function copies one file at a time. To copy multiple files with FlowForce, enclose the step which calls the copy function inside a For each step, as illustrated in the Copy Files example.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the Abort on error parameter, as follows:

- If the Abort on error parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see Handling Step Errors).
- If the Abort on error parameter is FALSE, the function returns FALSE.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>string as file</td>
<td>The path and file name of the source file that you want to copy.</td>
</tr>
<tr>
<td>Target</td>
<td>string as file</td>
<td>The path and file name of the destination directory. You can enter a different file name in the destination field if you want to rename it as well.</td>
</tr>
<tr>
<td>Overwrite</td>
<td>boolean</td>
<td>When true, causes the destination file to be overwritten. The default value is false.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If Abort on error is FALSE, the function will return Boolean FALSE as well. If Abort on error is TRUE, the job</td>
</tr>
</tbody>
</table>
Built-in Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td>Working directory</td>
<td>string as directory</td>
<td>Specifies the working directory (for example, c:somedirectory). If relative paths are used, they will be resolved against the working directory.</td>
</tr>
</tbody>
</table>

Examples
See Copy Files.

## 11.2.2 delete

Full path: /system/filesystem/delete

This function deletes a file from the path. When invoked from a simple execution step, this function deletes one file at a time. To delete multiple files with FlowForce, enclose the step which calls the delete function inside a For each step, as illustrated in the Copy Files example.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the Abort on error parameter, as follows:

- If the Abort on error parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see Handling Step Errors).
- If the Abort on error parameter is FALSE, the function returns FALSE.

Note: It is not possible for FlowForce to confirm directly from the delete function whether a file has been deleted. All FlowForce can do is get a response from the operating system that it is executing a delete command. If the job has subsequent steps that depend upon the deleted file, you will need to check explicitly whether the file still exists. You can you use the list-files function to check that.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>string as directory</td>
<td>The path and file name of the file you want to delete.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If Abort on error is FALSE, the function will return Boolean FALSE as well. If Abort on error is TRUE, the job</td>
</tr>
</tbody>
</table>
### 11.2.3 mkdir

**Full path:** /system/filesystem/mkdir

Creates a directory at the specified path.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see **Handling Step Errors**).  
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>string as directory</td>
<td>The path of the new directory.</td>
</tr>
<tr>
<td>Make parents</td>
<td>boolean</td>
<td>Select this check box to create a hierarchical path like c:\dir1\dir2\dir3 in one step.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If <strong>Abort on error</strong> is FALSE, the function will return Boolean FALSE as well. If <strong>Abort on error</strong> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td>Working directory</td>
<td>string as directory</td>
<td>Specifies the working directory (for example, c:somedirectory). If relative paths are used, they will be resolved against the working directory.</td>
</tr>
</tbody>
</table>
Examples
If **Working-Directory** is `c:\temp`, and **Path** is `temp2\temp3`, the function creates the new directory `c:\temp\temp2\temp3`.

11.2.4  **move**

Full path: `/system/filesystem/move`

Moves or renames a file.

When invoked from a simple execution step, this function moves or renames one file at a time. To move or rename multiple files with FlowForce, enclose the step which calls the move function inside a "for-each" step, similar to how this is done in the [Copy Files](#) example.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see [Handling Step Errors](#)).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>string as file</td>
<td>The path and file name of the source file that you want to move.</td>
</tr>
<tr>
<td><strong>Destination</strong></td>
<td>string as file</td>
<td>The name of the destination directory. If you supply only the directory name in this field, the original file name will be retained.</td>
</tr>
<tr>
<td><strong>Overwrite target</strong></td>
<td>boolean</td>
<td>When true, causes the destination file to be overwritten. The default value is false.</td>
</tr>
<tr>
<td><strong>Abort on error</strong></td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If <strong>Abort on error</strong> is FALSE, the function will return Boolean FALSE as well. If <strong>Abort on error</strong> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td><strong>Working directory</strong></td>
<td>string as directory</td>
<td>Specifies the working directory (for example, <code>c:\som edirectory</code>). If</td>
</tr>
</tbody>
</table>
# Built-in Functions / system/filesystem

## 11.2.5 rmdir

Full path: `/system/filesystem/rmdir`

Removes a directory.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see [Handling Step Errors](#)).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>string as directory</td>
<td>The name of the directory you want to delete.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If <strong>Abort on error</strong> is FALSE, the function will return Boolean FALSE as well. If <strong>Abort on error</strong> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td>Working directory</td>
<td>string as directory</td>
<td>Specifies the working directory (for example, <code>c:somedirectory</code>). If relative paths are used, they will be resolved against the working directory.</td>
</tr>
</tbody>
</table>
The `/system/ftp` container includes functions used to connect to an FTP or FTPS server, and perform operations such as uploading, retrieving, and deleting files, making or deleting remote directories, and others.

**Paths in FTP functions**
Some FTP functions have a Working Directory parameter. This parameter is common to many FlowForce functions. If you set the Working Directory, remember that all local paths supplied as parameters to the FTP function must be relative, not absolute. For example, when uploading a file, you can enter `C:\Upload` as Working Directory and `file.txt` as Source file. The final source path will become `C:\Upload\file.txt`.

In addition, some FTP functions have a Directory on host parameter that plays the same role as Working Directory, on the remote FTP server. Therefore, if you set the Directory on host, all remote paths supplied as parameters to the FTP function must be relative, not absolute. For example, when uploading a file, you can enter `uploads` as Directory on host and `file.csv` as Target file. The final target path will become `/uploads/file.csv`.

This is also important if you configured jobs as File System Triggers or HTTP Triggers. Such jobs have a triggerfile parameter that supplies the path of the file that triggered the job. If you intend to use the triggerfile parameter in any FTP function, remember that its path is absolute.

To obtain the file name with extension from the triggerfile, use the following FlowForce expression:

```
{filename-with-extension(triggerfile)}
```

For an example, see the FTP store function.

**Wildcards in FTP functions**
The following FTP functions accept wildcards as parameters:

- `/system/ftp/delete-wildcard`
- `/system/ftp/retrieve-wildcard`
- `/system/ftp/store-wildcard`

When using such functions, you can enter wildcards in the dedicated Wildcard parameter. The wildcards that you can use are as follows:

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>Usage</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>*</code></td>
<td>Match zero or more characters.</td>
<td><code>*.htm</code> will match <code>home.htm</code> and <code>index.htm</code></td>
</tr>
<tr>
<td><code>?</code></td>
<td>Match any single character.</td>
<td><code>*.xm?</code> will match <code>index.xml</code> and <code>project.xmi</code></td>
</tr>
</tbody>
</table>

The wildcard `+` (one or more) is not supported. Instead, you can use `?*` to achieve the same effect.
example, `*.c?*` will match `.cs`, `.cp` and `.csproj` files but will not match `.c` files.

### 11.3.1 delete

Full path: `/system/ftp/delete`

Deletes a file from the FTP server.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see *Handling Step Errors*).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

If you intend to use the **triggerfile** parameter in any FTP function, remember that its path is **absolute**. For an example, see the FTP **store** function.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address. Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, from which you want to delete a file. Optional parameter.</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see <em>Credentials</em>. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if...</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Use SSL/TLS encryption      | string  | (Optional parameter, the default value is `No`). To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to `No`. Otherwise, set this value to one of the following:  
  - Explicit with encrypted command channel  
  - Explicit with encrypted command and data channel  
  If you set any of the two options above, the server certificate will be validated according to the `Verify server certificate` parameter described below.  
  Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce. |
| Verify server certificate   | string  | (Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:  
  - No verification - Accept any certificate.  
  - Verify against system certificate store (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in `/usr/lib/ssl/cert.pem` and `/usr/lib/ssl/certs`, or the path where the `SSL_CERT_FILE` and... |
### Name: SSL_CERT_DIR
- **Type:** environment variables
- **Description:**
  - **Verify against selected server certificate** - FlowForce will compare the FTP server's certificate with the one specified in the `Server Certificate` parameter.

  Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.

### Name: Server certificate
- **Type:** certificate
- **Description:**
  - (Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to `Verify against selected server certificate`. Otherwise, this parameter value will be ignored.

### Name: Target file
- **Type:** string
- **Description:**
  - The name of the file that you want delete from the server.

  Mandatory parameter.

### Name: Abort on error
- **Type:** boolean
- **Description:**
  - This Boolean parameter determines what should be the return value of the function if the job fails. If `Abort on error` is FALSE, the function will return Boolean FALSE as well. If `Abort on error` is TRUE, the job execution is aborted. The default value is TRUE.

### Name: Account
- **Type:** string
- **Description:**
  - The FTP account name of the user allowed access to the files on the remote server.

  Optional parameter.
11.3.2 delete-wildcard

Full path: /system/ftp/delete-wildcard

Deletes from the FTP server any files that match a wildcard, for example, *.xml. Upon success, the function returns a list of deleted files (file name without path) or an empty list if no match was found. If execution fails, the outcome depends on the Abort on error parameter described below.

If you intend to use the triggerfile parameter in any FTP function, remember that its path is absolute. For an example, see the FTP store function.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address. Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, from which you want to delete a file. Optional parameter.</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see Credentials. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).</td>
</tr>
</tbody>
</table>
| Use SSL/TLS encryption | string   | (Optional parameter, the default value is No.) To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value }
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>to No. Otherwise, set this value to one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Explicit with encrypted command channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Explicit with encrypted command and data channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you set any of the two options above, the server certificate will be validated according to the Verify server certificate parameter described below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Implicit encryption is considered deprecated, and thus not supported in FlowForce.</td>
</tr>
<tr>
<td>Verify server certificate</td>
<td>string</td>
<td>(Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>No verification</strong> - Accept any certificate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Verify against system certificate store</strong> (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in <code>[/usr/lib/ssl/cert.pem</code> and <code>[/usr/lib/ssl/certs</code>, or the path where the <code>SSL_CERT_FILE</code> and <code>SSL_CERT_DIR</code> environment variables point to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Verify against selected server certificate</strong> - FlowForce will compare the FTP server's certificate with the one specified in the <strong>Server Certificate</strong></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter. Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.</td>
</tr>
<tr>
<td>Server certificate</td>
<td>certificate</td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to <code>Verify against selected server certificate</code>. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td>Wildcard</td>
<td>string</td>
<td>Mandatory parameter. Specifies a wildcard, for example, <code>*.xml</code>. Any files matching the wildcard will be deleted. See also <a href="#">Wildcards in FTP functions</a>.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This parameter dictates the function's behavior when execution fails. Namely, on execution failure, the function returns one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the <code>Abort on error</code> parameter is <code>false</code>, then the list of deleted files is not returned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the <code>Abort on error</code> parameter is <code>true</code>, then the function deletes files until failure is encountered, and then it aborts execution.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Therefore, some files may still be deleted even if execution fails.</td>
</tr>
<tr>
<td>Account</td>
<td>string</td>
<td>The FTP account name of the user allowed access to the files on the remote server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional parameter.</td>
</tr>
</tbody>
</table>
11.3.3  mkdir

Full path: /system/ftp/mkdir

Creates a directory on the FTP server.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see **Handling Step Errors**).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

If you intend to use the **triggerfile** parameter in any FTP function, remember that its path is absolute. For an example, see the FTP **store** function.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address. Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, where you want to create a new directory. Optional parameter.</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see <strong>Credentials</strong>. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).</td>
</tr>
<tr>
<td>Use SSL/TLS encryption</td>
<td>string</td>
<td>(Optional parameter, the default value is <strong>No</strong>.) To transfer</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to No. Otherwise, set this value to one of the following:</td>
</tr>
</tbody>
</table>
|                               |        | • Explicit with encrypted command channel  
|                               |        | • Explicit with encrypted command and data channel  
|                               |        | If you set any of the two options above, the server certificate will be validated according to the **Verify server certificate** parameter described below.                                                                                       |
|                               |        | Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce.                                                                                                                  |
| Verify server certificate    | string | (Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:                                                                                               |
|                               |        | • **No verification** - Accept any certificate.  
|                               |        | • **Verify against system certificate store** (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in /usr/lib/ssl/cert.pem and /usr/lib/ssl/certs, or the path where the SSL_CERT_FILE and SSL_CERT_DIR environment variables point to.  
<p>|                               |        | • <strong>Verify against selected server certificate</strong> - |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FlowForce will compare the FTP server’s certificate with the one specified in the Server Certificate parameter. Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.</td>
</tr>
<tr>
<td>Server certificate</td>
<td>certificate</td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to Verify against selected server certificate. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td>Target directory</td>
<td>string</td>
<td>The name of the directory that you want to create. Mandatory parameter.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If Abort on error is FALSE, the function will return Boolean FALSE as well. If Abort on error is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td>Account</td>
<td>string</td>
<td>The FTP account name of the user allowed access to the files on the remote server. Optional parameter.</td>
</tr>
</tbody>
</table>

### 11.3.4 move

Full path: /system/ftp/move
Moves a file on the FTP Server.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see **Handling Step Errors**).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

If you intend to use the **triggerfile** parameter in any FTP function, remember that its path is *absolute*. For an example, see the FTP **store** function.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address. Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, from where you want to move the file. Optional parameter.</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see <strong>Credentials</strong>. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).</td>
</tr>
<tr>
<td>Use SSL/TLS encryption</td>
<td>string</td>
<td>(Optional parameter, the default value is No.) To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to No. Otherwise, set this value to...</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Explicit with encrypted command channel</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Explicit with encrypted command and data channel</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you set any of the two options above, the server certificate will be validated according to the <code>Verify server certificate</code> parameter described below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce.</td>
</tr>
<tr>
<td><strong>Verify server certificate</strong></td>
<td><strong>string</strong></td>
<td>(Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>No verification</strong> - Accept any certificate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Verify against system certificate store</strong> (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in <code>/usr/lib/ssl/cert.pem</code> and <code>/usr/lib/ssl/certs</code>, or the path where the SSL_CERT_FILE and SSL_CERT_DIR environment variables point to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Verify against selected server certificate</strong> - FlowForce will compare the FTP server's certificate with the one specified in the <code>Server Certificate</code> parameter.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.</td>
</tr>
<tr>
<td>Server certificate</td>
<td>certificate</td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to <strong>Verify against selected server certificate</strong>. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td>Source file</td>
<td>string</td>
<td>The name of the file that you want to move.</td>
</tr>
<tr>
<td>Target file</td>
<td>string</td>
<td>The name of the copied file at the target location. Use a different name if you want to rename the copied file.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If <strong>Abort on error</strong> is FALSE, the function will return Boolean FALSE as well. If <strong>Abort on error</strong> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td>Account</td>
<td>string</td>
<td>The FTP account name of the user allowed access to the files on the remote server.</td>
</tr>
</tbody>
</table>

Optional parameter.
11.3.5 list

Full path: /system/ftp/list

Lists the contents of a directory on an FTP server. In case of successful execution, this function returns a sequence of string. Otherwise, the outcome depends on the Abort execution on error parameter, as further described below.

If you intend to use the triggerfile parameter in any FTP function, remember that its path is absolute. For an example, see the FTP store function.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address. Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, whose contents you want to list. Optional parameter. The default value is the current directory, &quot;/&quot;.</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see Credentials. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections). Optional parameter. The default value is true.</td>
</tr>
<tr>
<td>Use SSL/TLS encryption</td>
<td>string</td>
<td>(Optional parameter, the default value is No.) To transfer</td>
</tr>
</tbody>
</table>
### Built-in Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to <strong>No</strong>. Otherwise, set this value to one of the following:</td>
</tr>
</tbody>
</table>
|                             |          | - **Explicit with encrypted command channel**  
|                             |          | - **Explicit with encrypted command and data channel**                                                                                                                                                    |
|                             |          | If you set any of the two options above, the server certificate will be validated according to the **Verify server certificate** parameter described below.                                                                 |
|                             |          | Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce.                                                                                                                      |

<table>
<thead>
<tr>
<th><strong>Verify server certificate</strong></th>
<th><strong>string</strong></th>
<th>(Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:</th>
</tr>
</thead>
</table>
|                              |            | - **No verification** - Accept any certificate.  
|                              |            | - **Verify against system certificate store** (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in `/usr/lib/ssl/cert.pem` and `/usr/lib/ssl/certs`, or the path where the `SSL_CERT_FILE` and `SSL_CERT_DIR` environment variables point to.  
|                              |            | - **Verify against selected server certificate** - |

(Original text)
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>**FlowForce will compare the FTP server’s certificate with the one specified in the <strong>Server Certificate</strong> parameter. Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.</td>
</tr>
<tr>
<td><strong>Server certificate</strong></td>
<td>certificate</td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to <strong>Verify against selected server certificate</strong>. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td><strong>Wildcard</strong></td>
<td>string</td>
<td>The wildcard string to use, for example, &quot;.*.js&quot; if you need to retrieve all .js files from the directory specified by the <strong>Directory on host</strong> parameter. Optional parameter. The default value is an empty string, which means no wildcard filtering takes place.</td>
</tr>
<tr>
<td><strong>Abort on error</strong></td>
<td>boolean</td>
<td>This parameter controls what happens when execution has failed. If execution has failed and <strong>Abort on error is true</strong>, the job execution is aborted, and you can handle errors by means of <strong>protected blocks</strong>. If execution has failed and <strong>Abort on error is false</strong>, then the function returns an empty sequence. Optional parameter. The default value is true.</td>
</tr>
<tr>
<td><strong>Account</strong></td>
<td>string</td>
<td>The FTP account name of the user allowed access to the files on the remote server.</td>
</tr>
</tbody>
</table>
### Built-in Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional parameter.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 11.3.6 retrieve

Full path: `/system/ftp/retrieve`

Retrieves a file from the FTP Server.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see [Handling Step Errors](#)).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

If you intend to use the **triggerfile** parameter in any FTP function, remember that its path is **absolute**. For an example, see the FTP **store** function.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FTP Server</strong></td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address. Mandatory parameter.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td><strong>Directory on host</strong></td>
<td>string</td>
<td>The name of the directory, on the host, from where you want to retrieve the file. Optional parameter.</td>
</tr>
<tr>
<td><strong>Login credentials</strong></td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see <a href="#">Credentials</a>. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).</td>
</tr>
</tbody>
</table>
| Use SSL/TLS encryption        | string  | (Optional parameter, the default value is No.) To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to No. Otherwise, set this value to one of the following:
  - Explicit with encrypted command channel
  - Explicit with encrypted command and data channel
If you set any of the two options above, the server certificate will be validated according to the **Verify server certificate** parameter described below.
Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce. |
| Verify server certificate      | string  | (Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:
  - **No verification** - Accept any certificate.
  - **Verify against system certificate store** (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in `/usr/lib/ssl/cert.pem` and `/usr/lib/ssl/certs`, or... |
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source file</td>
<td>string</td>
<td>The name of the file that you want to retrieve. Mandatory parameter.</td>
</tr>
<tr>
<td>Target file</td>
<td>string</td>
<td>The name the file should have once it is retrieved. Mandatory parameter.</td>
</tr>
<tr>
<td>Overwrite target</td>
<td>boolean</td>
<td>When true, causes the destination file to be overwritten. The default value is false.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If Abort on error is FALSE, the function will return Boolean FALSE as well. If Abort on error is TRUE, the job</td>
</tr>
</tbody>
</table>
### 11.3.7 retrieve-wildcard

Full path: /system/ftp/retrieve-wildcard

Retrieves from the FTP server any files that match a wildcard, for example, *.xml. Upon success, the function returns a list of written files (absolute local paths) or an empty list, if no match has been found. If execution fails, the outcome depends on the **Abort on error** parameter described below.

If you intend to use the **triggerfile** parameter in any FTP function, remember that its path is **absolute**. For an example, see the FTP **store** function.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>FTP Server</td>
</tr>
<tr>
<td>Port</td>
</tr>
<tr>
<td>Directory on host</td>
</tr>
<tr>
<td>Login credentials</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>-------------------------------------</td>
</tr>
</tbody>
</table>
|                                     |               | credential record, see Credentials.
|                                     |               | Skip this parameter if the FTP server does not require credentials.                                                                                                                                                                                                       |
|                                     | boolean       | Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).                                                                                                                                               |
|                                     | string        | (Optional parameter, the default value is No.) To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to No. Otherwise, set this value to one of the following: |
|                                     |               | - Explicit with encrypted command channel
|                                     |               | - Explicit with encrypted command and data channel
|                                     |               | If you set any of the two options above, the server certificate will be validated according to the Verify server certificate parameter described below.                                                                                                                   |
|                                     |               | Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce.                                                                                                                                                                                   |
|                                     | string        | (Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:                                                                                                                                                                    |
|                                     |               | - No verification - Accept any certificate.
|                                     |               | - Verify against system certificate store (default value) - On Windows, use the certificate store of the user account running the
<p>| | | |
|                                     |               |                                                                                                                                                                                                                                                                                           |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in /usr/lib/ssl/cert.pem and /usr/lib/ssl/certs, or the path where the SSL_CERT_FILE and SSL_CERT_DIR environment variables point to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Verify against selected server certificate - FlowForce will compare the FTP server's certificate with the one specified in the Server Certificate parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.</td>
</tr>
<tr>
<td>Server certificate</td>
<td>certificate</td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to Verify against selected server certificate. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td>Wildcard</td>
<td>string</td>
<td>Mandatory parameter. Specifies a wildcard, for example, *.xml. Any files matching the wildcard will be retrieved. See also Wildcards in FTP functions.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This parameter dictates the function's behavior when execution fails. Namely, on execution failure, the function returns one of the following:</td>
</tr>
</tbody>
</table>
### Built-in Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort on error</td>
<td></td>
<td>If the <strong>Abort on error</strong> parameter is <strong>false</strong>, then the list of retrieved files is not returned. If the <strong>Abort on error</strong> parameter is <strong>true</strong>, then the function retrieves files until failure is encountered, and then it aborts execution. Therefore, some files may still be retrieved even if execution fails.</td>
</tr>
<tr>
<td>Working directory</td>
<td><code>string</code></td>
<td>The directory where all files retrieved from the FTP server should be stored.</td>
</tr>
<tr>
<td>Account</td>
<td><code>string</code></td>
<td>The FTP account name of the user allowed access to the files on the remote server. Optional parameter.</td>
</tr>
</tbody>
</table>

### 11.3.8 `rmdir`

**Full path:** `/system/ftp/rmdir`

Deletes a directory from the FTP server.

This function returns Boolean **TRUE** if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is **TRUE** (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see Handling Step Errors).
- If the **Abort on error** parameter is **FALSE**, the function returns **FALSE**.

If you intend to use the **triggerfile** parameter in any FTP function, remember that its path is **absolute**. For an example, see the FTP **store** function.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td><code>string</code></td>
<td>Address of the remote FTP server, either as a URL or IP address.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, from where you want to delete the directory. Optional parameter.</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see Credentials. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).</td>
</tr>
<tr>
<td>Use SSL/TLS encryption</td>
<td>string</td>
<td>(Optional parameter, the default value is No.) To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to No. Otherwise, set this value to one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Explicit with encrypted command channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Explicit with encrypted command and data channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you set any of the two options above, the server certificate will be validated according to the Verify server certificate parameter described below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Implicit encryption is considered deprecated, and thus not supported in 139.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Verify server certificate</strong></td>
<td><strong>string</strong></td>
<td>(Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>No verification</strong> - Accept any certificate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Verify against system certificate store</strong> (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in /usr/lib/ssl/cert.pem and /usr/lib/ssl/certs, or the path where the SSL_CERT_FILE and SSL_CERT_DIR environment variables point to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Verify against selected server certificate</strong> - FlowForce will compare the FTP server's certificate with the one specified in the Server Certificate parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.</td>
</tr>
<tr>
<td><strong>Server certificate</strong></td>
<td><strong>certificate</strong></td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to <strong>Verify against selected server certificate</strong>. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Target directory</td>
<td>string</td>
<td>The name the directory that you want to delete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mandatory parameter.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If <strong>Abort on error</strong> is FALSE, the function will return Boolean FALSE as well. If <strong>Abort on error</strong> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td>Account</td>
<td>string</td>
<td>The FTP account name of the user allowed access to the files on the remote server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional parameter.</td>
</tr>
</tbody>
</table>

### 11.3.9 store

Full path: `/system/ftp/store`

Uploads a file to the FTP server.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see Handling Step Errors).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, where you want to store the file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional parameter.</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see <a href="#">Credentials</a>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).</td>
</tr>
</tbody>
</table>
| Use SSL/TLS encryption    | string     | (Optional parameter, the default value is No.) To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to No. Otherwise, set this value to one of the following:  
|                           |            |  
|                           |            | * Explicit with encrypted command channel  
|                           |            | * Explicit with encrypted command and data channel  
|                           |            | If you set any of the two options above, the server certificate will be validated according to the [Verify server certificate](#) parameter described below.  
<p>|                           |            | Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce. |
| Verify server certificate  | string     | (Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values: |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No verification</td>
<td></td>
<td>• <strong>No verification</strong> - Accept any certificate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Verify against system certificate store</strong> (default value) - On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in <code>/usr/lib/ssl/cert.pem</code> and <code>/usr/lib/ssl/certs</code>, or the path where the <code>SSL_CERT_FILE</code> and <code>SSL_CERT_DIR</code> environment variables point to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>Verify against selected server certificate</strong> - FlowForce will compare the FTP server's certificate with the one specified in the <code>Server Certificate</code> parameter.</td>
</tr>
<tr>
<td>Server certificate</td>
<td>certificate</td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to <strong>Verify against selected server certificate</strong>. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td>Source file</td>
<td>string</td>
<td>The name of the file to be uploaded to the FTP Server. Mandatory parameter.</td>
</tr>
</tbody>
</table>
### Name | Type | Description
--- | --- | ---
Target file | string | The name the file should have once it is uploaded to the FTP Server. This can be different from the Source File. Mandatory parameter.
Abort on error | boolean | This Boolean parameter determines what should be the return value of the function if the job fails. If Abort on error is FALSE, the function will return Boolean FALSE as well. If Abort on error is TRUE, the job execution is aborted. The default value is TRUE.
Working directory | string | Specifies the working directory of the job (for example, c:\somedirectory). If relative paths are used, they will be resolved against the working directory.
Account | string | The FTP account name of the user allowed access to the files on the remote server. Optional parameter.

### Examples
This example shows you how to upload a file to a remote FTP server, without knowing the file name and extension at job configuration time. This is possible by configuring the job to run as a file system trigger, as shown below:

The trigger above monitors the directory C:\FlowForce\Upload for changes. Whenever you copy a file to this directory, the job fires, and the absolute path of the file that triggered the job becomes available in the triggerfile input parameter. This enables you to use this file in the job without knowing its name and extension, as described below.
In the job configuration above, the **store** function is called with the following parameters:

- **FTP Server** - The address of the FTP server (an I.P. address, in this example)
- **Port** - The default port 21
- **Directory on host** - In this example, we would like all uploaded files to be put in the "uploads" subdirectory on the server, relative to the FTP root directory.
- **Login credentials** - The FTP username and password required to connect to the FTP server. For the sake of reuse, these were previously defined as credentials and here are just referenced from the **public** container.
- **Source file** - The path of the local file to be uploaded. In this example, this must be a relative path, because **Working directory** is set, see below.
- **Target file** - The path of the file on the FTP server after upload. In this example, this must also be a relative path, because **Directory on host** is set, see below.
- **Working directory** - A directory on the local computer. All local relative file paths are assumed to be relative to this directory. Notice that it is the same as the polling directory defined in the trigger.

If **Working directory** is set, **Source file** must be a relative, not absolute, path. Likewise, if **Directory on host** is set, the **Target file** must be a relative path.
This example uses both Working Directory and Directory on host; therefore, we need to convert the absolute path of the triggerfile to relative.

To achieve this, Source file uses a FlowForce expression. This expression takes the triggerfile as argument (recall that this is an absolute path), and returns just the file name and extension. For example, if triggerfile is C:\data.txt, the expression would return just data.txt. The same happens with the expression in the Target file. For more details about expressions in FlowForce, see The FlowForce Expression Language.

With the configuration above, the following happens whenever you copy a file (regardless of its extension) to the working directory:

- Assuming that you've copied a file called data.txt, the job fires and gets C:\FlowForce\Upload\data.txt as triggerfile.
- Thanks to the expression, Source File becomes data.txt, and so does the Target file.
- The actual path of the file to upload is obtained by concatenating the Working directory with the Source File.
- The destination path of the file on the server is obtained by concatenating the Directory on host with the Target file.
- FlowForce attempts to connect with the supplied FTP credentials. On success, it puts the file data.txt in the uploads directory on the FTP server.

### 11.3.10 store-wildcard

Full path: /system/ftp/store-wildcard

Uploads to the FTP server files from a local directory, if they match a wildcard, for example, *.xml. Upon success, the function returns a list of uploaded files (absolute local paths) or an empty list, if no match has been found. If execution fails, the outcome depends on the Abort on error parameter described below.

If you intend to use the triggerfile parameter in any FTP function, remember that its path is absolute. For an example, see the FTP store function.

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Server</td>
<td>string</td>
<td>Address of the remote FTP server, either as a URL or IP address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mandatory parameter.</td>
</tr>
<tr>
<td>Port</td>
<td>number</td>
<td>The port number used to connect to the FTP server. The default value is 21.</td>
</tr>
<tr>
<td>Directory on host</td>
<td>string</td>
<td>The name of the directory, on the host, from which you want to delete a file.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>Login credentials</td>
<td>credential</td>
<td>The username and password of the FTP account, as a FlowForce credential record, see Credentials. Skip this parameter if the FTP server does not require credentials.</td>
</tr>
<tr>
<td>Use passive mode</td>
<td>boolean</td>
<td>Use passive mode if connection problems occur (for example, if routers or firewalls are set up to prevent active connections).</td>
</tr>
<tr>
<td>Use SSL/TLS encryption</td>
<td>string</td>
<td>(Optional parameter, the default value is No.) To transfer information, FTP uses a command channel and a data channel. If you would like to transfer FTP data without encryption, set this value to No. Otherwise, set this value to one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Explicit with encrypted command channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Explicit with encrypted command and data channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you set any of the two options above, the server certificate will be validated according to the Verify server certificate parameter described below.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Implicit encryption is considered deprecated, and thus not supported in FlowForce.</td>
</tr>
<tr>
<td>Verify server certificate</td>
<td>string</td>
<td>(Optional parameter.) Specifies how FlowForce should verify the FTP server's certificate. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No verification - Accept any certificate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Verify against system certificate store (default</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>value)</td>
<td></td>
<td>- On Windows, use the certificate store of the user account running the job and the system store to verify the certificate signature. On Linux, use the system certificate store, usually located in <code>/usr/lib/ssl/cert.pem</code> and <code>/usr/lib/ssl/certs</code>, or the path where the <code>SSL_CERT_FILE</code> and <code>SSL_CERT_DIR</code> environment variables point to.</td>
</tr>
<tr>
<td>value)</td>
<td></td>
<td>- <strong>Verify against selected server certificate</strong> - FlowForce will compare the FTP server's certificate with the one specified in the <code>Server Certificate</code> parameter.</td>
</tr>
<tr>
<td>value)</td>
<td></td>
<td>Using this parameter requires the presence of a server certificate and a secure connection. If a secure connection cannot be established, the FTP function will fail.</td>
</tr>
<tr>
<td>Server certificate</td>
<td>certificate</td>
<td>(Optional parameter.) Specifies the path to a certificate object in FlowForce. The specified FlowForce certificate will be verified against the FTP server certificate if you also set the previous parameter to <strong>Verify against selected server certificate</strong>. Otherwise, this parameter value will be ignored.</td>
</tr>
<tr>
<td>Wildcard</td>
<td>string</td>
<td>Mandatory parameter. Specifies a wildcard, for example, <code>*.xml</code>. Any files from the directory specified by the <code>Working Directory</code> parameter will be uploaded if they match this wildcard.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See also <a href="#">Wildcards in FTP functions</a>.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This parameter dictates the function's behavior when execution fails. Namely, on execution failure, the function returns one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the <strong>Abort on error</strong> parameter is <strong>false</strong>, then the list of uploaded files is not returned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the <strong>Abort on error</strong> parameter is <strong>true</strong>, then the function uploads files until failure is encountered, and then it aborts execution.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Therefore, some files may still be uploaded even if execution fails.</td>
</tr>
<tr>
<td>Working directory</td>
<td>string</td>
<td>The directory from which files are to be uploaded to the FTP server, if they match the wildcard.</td>
</tr>
<tr>
<td>Account</td>
<td>string</td>
<td>The FTP account name of the user allowed access to the files on the remote server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional parameter.</td>
</tr>
</tbody>
</table>
11.4 /system/mail

The /system/mail container includes the functions `send` and `send-mime`, used to send email.

11.4.1 send

Full path: /system/mail/send

Sends e-mail to the specified recipients, generally the administrator.

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see *Handling Step Errors*).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

Before using this function, ensure that the mail server settings are configured (see Setting Mail Parameters).

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| From       | string | Email address from which the e-mail message is to be sent, for example: `flowforce@<hostname>`.
| To         | string | Recipient's email address. Mandatory parameter. This field may also contain a comma-separated list of multiple destination e-mail addresses. |
| Subject    | string | Subject line of the message. Mandatory parameter. |
| Message body | string | Optional parameter that provides the body text of the message, as string. The message body supports ASCII as well as Unicode characters. The text box for the message body allows entering multiple lines and expressions in curly { } braces. |
### Built-in Functions

**Name** | **Type** | **Description**
--- | --- | ---
Attachment | string as file | File name of the attachment sent with the email.  
Abort on error | boolean | This Boolean parameter determines what should be the return value of the function if the job fails. If **Abort on error** is FALSE, the function will return Boolean FALSE as well. If **Abort on error** is TRUE, the job execution is aborted. The default value is TRUE.

### Examples

See [Adding Error Handling to a Job](#).

#### 11.4.2 send-mime

**Full path:** `/system/mail/send-mime`

Sends e-mail to the specified recipients, generally the administrator.

> Before using this function, ensure that the mail server settings are configured (see Setting Mail Parameters).

This function returns Boolean TRUE if execution was successful. If the job execution fails, the outcome depends on the value of the **Abort on error** parameter, as follows:

- If the **Abort on error** parameter is TRUE (default value), the job execution is aborted. In this case, you can still handle errors by means of protected blocks (see Handling Step Errors).
- If the **Abort on error** parameter is FALSE, the function returns FALSE.

Unlike the **send** function, the **Message body** parameter of this function expects an expression that produces a stream, not a string. This enables you to get the message body (for example, as HTML) from a stream.

> To obtain HTML content for the message body, it is strongly recommended to call a StyleVision Server transformation that produces HTML output as MIME. FlowForce Server by itself does not collect any images, stylesheets, or similar resources referenced by HTML files into a self-contained MIME stream.

In order for produce a self-contained HTML message body with StyleVision Server, do the following:

1. Design the HTML body of the email in Altova StyleVision. The design may contain local images and stylesheets.
2. Deploy the StyleVision transformation to FlowForce Server. In FlowForce, the transformation becomes a built-in FlowForce function that can be executed by StyleVision Server.
3. Create a job that calls the StyleVision Server transformation above, making sure to select the **GenerateHtmlOutputAsMime** option in the job configuration page.

4. In the job configuration page, call FlowForce Server expression functions to pick up the generated MIME stream and pass it to the "Message body" parameter of the `send-mime` function (see “Example 1” below).

   If any external resources referenced by the HTML file cannot be embedded into the MIME stream, they will be added as attachments to the email.

An example job that produces HTML output as a MIME stream is illustrated below. For a step-by-step example that illustrates how to deploy StyleVision transformation to FlowForce Server, see Creating a Job from a StyleVision Transformation. For more information about StyleVision Server integration, see Integration with Other Altova Servers.

To create the stream for the message body directly in FlowForce, you can also call expression functions such as `stream-open` or `stream-from-string`. Likewise, you can use MIME expression functions to customize the e-mail or attachment message headers.

   To prevent the e-mail from landing into the “Junk” folder on the recipient's side, you should construct the MIME headers in a way that is allowed by the receiving server or program.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>string</td>
<td>Email address from which the e-mail message is to be sent, for example: <code>flowforce@&lt;hostname&gt;</code>.</td>
</tr>
<tr>
<td>To</td>
<td>string</td>
<td>Recipient's email address. Mandatory parameter. This field may also contain a comma-separated list of multiple destination e-mail addresses.</td>
</tr>
<tr>
<td>Subject</td>
<td>string</td>
<td>Subject line of the message. Mandatory parameter.</td>
</tr>
<tr>
<td>Message body</td>
<td>stream</td>
<td>Body text of the message, as a FlowForce expression that returns a stream type.</td>
</tr>
<tr>
<td>Attachment</td>
<td>sequence of stream</td>
<td>The attachment(s) sent with the email. Each attachment must be a FlowForce expression that produces a stream. Call <code>stream expression functions</code> to create streams from strings or files. Call MIME expression functions to...</td>
</tr>
</tbody>
</table>
### Built-in Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>add, modify, or delete MIME headers.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If <strong>Abort on error</strong> is FALSE, the function will return Boolean FALSE as well. If <strong>Abort on error</strong> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
</tbody>
</table>

### Examples

**Example 1:** The job illustrated below invokes the `send-mime` function in order to send an e-mail in HTML format.

The first execution step generates HTML output by calling a StyleVision Server transformation. This transformation was designed with StyleVision and then deployed to FlowForce Server, as illustrated in [Creating a Job from a StyleVision Transformation](#). Notice that the *GenerateHtmlOutputAsMime* check box is selected. Also, the result of the first execution step was called "output" (we will need this string in a subsequent step and will refer to it simply by typing "output" inside an expression).

The second execution step calls the `compute` function in order to compute a FlowForce expression. Namely, the expression `nth(results(output), 0)` picks up the MIME stream with index 0 (in this case, *OutHtml*) from the result generated by the previous step.

Finally, the third execution step sends the actual e-mail message. The "Message body" field also contains a FlowForce expression, which in this case is the result computed previously in step 2.
Example 2: The job illustrated below invokes the `send-mime` function in order to send an e-mail in HTML format that also contains an image attachment in `.png` format.
The first execution step prepares the HTML code for the message body. For simplicity, the HTML code in this example was typed directly in the text box. The recommended way to obtain HTML output is to call a StyleVision Server transformation, as illustrated in the previous example. This execution step produces some string output that will be used in a subsequent step.

The second execution step produces the body of the e-mail, as a stream. Namely, it converts the result of the first execution step (\texttt{body.html}, of type \texttt{string}) to a stream. The \texttt{stream-from-string} function was called for that purpose. The desired encoding and MIME type are supplied as arguments to the function.

The third execution step creates the attachment of the e-mail, also as a stream. The image attachment is from a local path, \texttt{C:\sample.png}. More specifically, this step computes the following expression:

\begin{verbatim}
add-mime-header(stream-open('C:\sample.png', 'image/png'), 'Content-Disposition', 'attachment; filename=sample.png')
\end{verbatim}

The expression above does the following:

- The \texttt{stream-open} function opens the image as a stream
- The \texttt{add-mime-header} function adds the "Content-Disposition" header to the stream. This way, the image attachment will have the intended name ("sample.png").

Finally, the expression from the "Attachment" field is required because this is a binary file. Such files have to be encoded as \texttt{base-64} in order to be preserved during transmission, by using the \texttt{mime-content-encode} expression function.
11.5 /system/maintenance

The /system/maintenance container includes functions used to perform maintenance operations on the server.

11.5.1 archive-log

Full path: /system/maintenance/archive-log

Moves the older log records to an archive file on the server. Returns the name of the archive file that was created, as string value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Older than, days</td>
<td>number</td>
<td>Archives files older than the number of days entered here. The default value is 30.</td>
</tr>
<tr>
<td>Archive directory</td>
<td>string</td>
<td>Archive directory name, (for example, c:\temp). Mandatory.</td>
</tr>
<tr>
<td>Archive file prefix</td>
<td>string</td>
<td>Specifies the prefix of the archive file. The default value is flowforcelog.</td>
</tr>
<tr>
<td>Delete archived records</td>
<td>boolean</td>
<td>Select this check box to delete archived records from the FlowForce Server database.</td>
</tr>
<tr>
<td>Working directory</td>
<td>string</td>
<td>Specifies the working directory of the job (for example, c:\somedirectory). If relative paths are used, they will be resolved against the working directory.</td>
</tr>
</tbody>
</table>

11.5.2 cleanup-files

Full path: /system/maintenance/cleanup-files

Deletes those files that are not in use or referenced by any deployed objects (such as MapForce mappings and StyleVision transformations). Returns the number of files that were deleted, as numeric value.

When you delete deployed objects, or when you re-deploy existing objects with modified files, any files associated with previously deployed objects become unused. By default, FlowForce Server does not delete the unused files. Therefore, in order to clean up the disk space, it is strongly recommended to create a job which periodically calls this function, especially in enterprise environments where multiple users deploy objects to FlowForce Server.
To see the current disk space used by deployed objects, check the size of the files folder located in the FlowForce Server application data folder (see FlowForce Server Application Data).

This function does not have any parameters.

11.5.3 truncate-log

Full path: /system/maintenance/truncate-log

Deletes log records older than the date supplied. Returns the number of records that were deleted, as numeric value.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Older than, days</td>
<td>number</td>
<td>Truncates (deletes) records older than the number of days entered here. The default value is 30.</td>
</tr>
</tbody>
</table>
11.6 /system/shell

The /system/shell container includes the function commandline, used to execute shell commands or scripts.

11.6.1 commandline

Full path: /system/shell/commandline

Executes a shell command or a batch file.

To have FlowForce Server jobs read environment variables, they must be defined in scripts, and those scripts must be executed with the /system/shell/commandline function. Be aware that FlowForce Server is running a non-interactive shell, which means all behavior specific to interactive shells is not applicable (such as executing .profile or .bashrc on Linux).

If the exit code from the last shell command is other than "0", the outcome is as follows:

- If the parameter Abort on error is true (default), this function aborts execution. In this case, you can handle the error by means of protected blocks (see Handling Step Errors).
- If the parameter Abort on error is false, the function returns the result of the shell command, including the standard output, the standard error, and the exit code.

If the exit code from the last command is "0" (success), the function returns the result of the last shell command, as generic type. To handle the value returned by this function in another step or job, do the following:

1. Name the returned result by entering a value in the Assign this step’s result to text box (for example, "myresult").
2. Create a new step which executes either the function compute or compute-string, depending on what return type you need.
3. Enter as argument to the above function an expression which gets the desired part from the generic result. For example, enter the expression stdout(myresult) to get the standard output of the result as stream, and stderr(myresult) to get the standard error output stream. To get the same values as string, use content(stdout(myresult)) and content(stderr(myresult)), respectively.

Note that the stdout function (and the job) will fail if the shell command does not return a standard output. Likewise, the stderr function will fail if there is no standard error.

See also Handling Data Types in Steps and Step Result Functions.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>string</td>
<td>Enter the shell command to execute.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>boolean</td>
<td>This Boolean parameter</td>
</tr>
</tbody>
</table>
### Name | Type | Description
--- | --- | ---
 |  | determines what should be the return value of the function if the job fails. If \texttt{Abort on error} is FALSE, the function will return Boolean FALSE as well. If \texttt{Abort on error} is TRUE, the job execution is aborted. The default value is TRUE.

| Working directory | string as directory | Specifies the working directory of the job (for example, \texttt{c:\some directory}). If relative paths are used, they will be resolved against the working directory. |

#### Examples

The following job executes a Windows batch file called \texttt{DoTransform.bat}. Assuming that the \texttt{DoTransform.bat} requires some XML file as input, the input XML file must be copied to the working directory. In this example, the working directory is \texttt{C:\codegen\xslt2}.

The following job calls RaptorXML Server to run an XSLT transformation with parameters. It is assumed that the \texttt{PATH} environment variable contains the path to the RaptorXML Server executable, for example \texttt{C:\Program Files (x86)\Altova\RaptorXMLServer2022\bin}. For more information about RaptorXML Server, see \url{https://www.altova.com/raptorxml}.

For a step-by-step example which handles the output returned by the command line, see Check if a path exists.
12 Expression Functions

This chapter provides reference to the FlowForce expression functions. To understand how to use expressions, see FlowForce Expressions 336. The list of available expression functions is given below:

- Step Result Functions 336
- Stream Functions 346
- File System Functions 349
- File Path Functions 352
- List Functions 357
- String Functions 364
- Boolean Functions 372
- Runtime Information Functions 375
- MIME Expression Functions 378
12.1 Step Result Functions

Step result functions allow you to process the result returned by jobs (or the result returned by execution steps within jobs).

12.1.1 error-message

This function returns the text of the error message encountered by a step. The typical usage of this function is inside a protected block, and specifically inside the "On Error" handler. The function may return an empty string if no error has been encountered or if it is not technically possible to retrieve the text of the error due to the nature of the job.

Signature

\[ \text{error-message}(\text{result: result}) \rightarrow \text{string} \]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>result</td>
<td>Supplies the erroneous step from which the error text should be retrieved. To get the erroneous step, call the failed-step() function.</td>
</tr>
</tbody>
</table>

Examples

See Add error handling to a job.


12.1.2 exitcode

Returns the numeric exit code of the result.

Signature

```
exitcode(result: result) -> number
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>result</td>
<td>The result of the step whose exit code you want to get.</td>
</tr>
</tbody>
</table>

Examples

The following job consists of two steps. The first step invokes a Windows command line command which attempts to create a directory called **data** in the current working directory (**C:\FlowForce**). The result of this step is declared as **outcome**. The second step gets the **outcome** and returns the numeric exit code from it, with the help of the **exitcode** function. The numeric exit is then converted to a string, with the help of the **string** function. This conversion is required because the data type of the expression is string.

Importantly, the **Abort on error** option is not selected; otherwise, the execution would stop in case of error, and so there wouldn't be any exit code for the second step to process.

When the job runs for the first time, the **data** directory is supposedly created successfully, and the exit code would be 0. On subsequent runs, it cannot be created because it already exists, so the exit code would be 1.

See also **Adding Error Handling to a Job**.
12.1.3 failed-step

Returns the result of a failed execution step. Using this function is meaningful when you are handling errors with protected blocks, as described in Handling Step Errors. The failed-step function must be part of the "On error" handler; otherwise, the step where you are using it will fail because there is no erroneous step.

This function returns a value of type result that represents the result of the erroneous step. To find the result's attributes, pass this function as argument to expression functions such as stdout or stderr, for example:

```
stderr(failed-step())
stdout(failed-step())
```

Whether you should use stderr or stdout depends on whether the failing step returns the error information in the standard error or standard output streams, respectively.

**Signature**

```
failed-step() -> result
```

**Examples**

The job illustrated below uses error handling, so it qualifies for a call to the failed-step function. The first execution step attempts to run a shell command which is supplied as a job input parameter. If the command fails with an error, the "On error" handler will be executed. The first and only step of the "On Error" handler calls an error handling sub-job which was created separately and is discussed below.
Although you can configure your error handling differently, the error-handling sub-job in this example takes two input parameters:

1. `inputResult` - the output of the `failed-step` function, of type `result`.
2. `workingDirectory` - the directory to which the log file containing the error details will be written.

The error handling sub-job looks as follows:
The execution step above invokes the `copy` function in order to create a file called `error.log` in the job's working directory. The expression from the `Source` text box does the following:

1. The `stderr` expression function converts the standard error provided by `inputResult` to a stream. As mentioned above, in some cases, you might need to use `stdout` instead of `stderr`. Both `stdout` and `stderr` take a value of type `result` as argument. That's precisely the return type produced by the `failed-step` function (which in this example was called in the main job).
2. The `as-file` function converts the stream to a file and writes it to the disk. The path of the file is relative to the working directory.

For more examples, see:

- Add Error Handling to a Job
- Validate an XML Document with Error Logging

### 12.1.4 `results`

Returns an array of streams of the specified result, optionally filtered by name. Use the function `nth` to access a particular value in the array.

**Signature**

```
results(result: result, name: string) -> array of stream
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>result</td>
<td>Mandatory parameter. The result of the step from which you want to return an array of streams.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Optional parameter. When provided, filters by name a particular value in the result.</td>
</tr>
</tbody>
</table>

Examples

Example A. Let’s suppose that you have deployed to FlowForce Server a mapping that generates a single XML file as output. An example of such a mapping is **CompletePO.mfd** included with MapForce examples. The name of the target XML component in MapForce is “CompletePO”. To process the result of this mapping and save it to a file from FlowForce, configure the job as follows:

In the job configuration above, the first step runs the mapping and returns the result as **mapping_result**. In the second step, the expression

```
{as-file(nth(results(mapping_result), 0))}
```
processes the **mapping_result** and converts it to a file. Namely, the **results** function picks the array of streams from the MapForce component. The **nth** function picks the first item from this array. Finally, the **as-file** function generates a file from the stream.

The **copy** function copies the generated file to the working directory. The **Target** text box defines the name of the generated file. Any existing file with the same name will be overwritten.

*Example B.* Let's suppose that you have deployed to FlowForce Server a mapping that has two target XML components, "MarketingExpenses" and "DailyExpenses". An example of such a mapping is **MarketingAndDailyExpenses.mfd** included with MapForce examples. To generate a file from the "DailyExpenses" component, create a job similar to the one above, but change the expression to:

```
{as-file(nth(results(mapping_result,'DailyExpenses'), 0))}
```

The only difference here is that the array of streams produced by the mapping is filtered by the name of the desired component (in this case, "DailyExpenses").

*Example C.* Let's suppose that you have deployed to FlowForce Server a mapping that generates multiple XML files dynamically. The output file names are generated by the mapping itself and are not known before runtime. An example of such a mapping is **DividePersonsByDepartmentIntoGroups.mfd** included with MapForce examples. To generate the third output file of the mapping, create a job similar to the one above, and change the expression to:

```
{as-file(nth(results(mapping_result), 2))}
```

Here we need the third file, so the index supplied as second argument to the nth function is 2 (not 3), because the index is zero-based.

See also the following examples:

- [Creating a Job from a StyleVision Transformation](#)
- [Generate Multiple PDFs from Multiple XMLs](#)

### 12.1.5 retry-count

Returns a number that indicates how many times FlowForce re-tries the execution of one or more steps that have error/success handling (a so-called "protected block"). Note that the function specifically evaluates the innermost protected block surrounding the function. If no retries took place (that is, if the first run of the protected block was successful), the return value is 0. See also [Retry on Error](#).

**Signature**

```
retry-count() -> number
```
12.1.6  stdout

Some execution steps (such as those that run shell commands) return standard output. For example, the shell command `dir` (on Windows) returns a list of directories.

When a step returns a result, FlowForce Server automatically assigns to it the generic type `result`. With the `stdout` function, you can get access to the standard output of `result`, as follows:

```plaintext
stdout(result)
```

where `result` is the value returned by some execution step.

This function fails if `result` does not provide standard output.

**Signature**

```plaintext
stdout(result: result) -> stream
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>result</td>
<td>The result of the step whose standard error you want to get.</td>
</tr>
</tbody>
</table>

**Examples**

See the following examples:

- Adding Error Handling to a Job
- Validate an XML Document with Error Logging
- Check if a Path Exists
12.1.7 stderr

Returns the standard error of the result. Fails if the result does not provide a standard error.

Signature

stderr(result: result) -> stream

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>result</td>
<td>The result of the step whose standard error you want to get.</td>
</tr>
</tbody>
</table>

Examples

See Adding Error Handling to a Job for an example.
12.2 Stream Functions

Stream functions are used to process streams of data. You can pass streams to FlowForce Server either by means of Web services or from step results.

12.2.1 as-file

Creates a file if the stream source is a file. Creates a temporary file if the stream source is not a file.

Signature

\[
\text{as-file} : \text{stream} -> \text{string}
\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>stream</td>
<td>Specifies the stream source.</td>
</tr>
</tbody>
</table>

Examples

The following job creates a file called file.txt with one line of text. First, the stream-from-string function generates a stream from the text supplied as argument. Next, the as-file function takes the stream as argument and generates a temporary file from it. To copy the temporary file to a permanent path, the built-in copy function is called from a separate execution step. The file is copied to the working directory of the job (C:\FlowForce) and will be overwritten each time the job runs.

Execution Steps

See also Validate an XML Document with Error Logging.
12.2.2  content

Reads the contents of the specified stream as text in the specified encoding.

Signature

```
content(stream: stream, encoding: string="UTF-8") -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stream</td>
<td>stream</td>
<td>Specifies the stream source.</td>
</tr>
<tr>
<td>encoding</td>
<td>string</td>
<td>Specifies the encoding to use. The default encoding is 'UTF-8'.</td>
</tr>
</tbody>
</table>

Examples

See the following example:

- Adding Error Handling to a Job

12.2.3  empty-stream

Creates an empty stream.

Signature

```
empty-stream() -> stream
```

12.2.4  stream-from-string

Creates a stream from a string using the supplied encoding. The content type supplied as argument is associated to the stream. This type of stream is not automatically saved as a file.

Signature

```
stream-from-string(string: string, encoding: string="UTF-8", contenttype: string=contenttype=text/plain) -> stream
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The string from which the stream should be created.</td>
</tr>
<tr>
<td>encoding</td>
<td>string</td>
<td>Specifies the encoding to use. The default encoding is 'UTF-8'.</td>
</tr>
<tr>
<td>contenttype</td>
<td>string</td>
<td>Specifies the <strong>contenttype</strong> to associate to the stream. The default is <code>contenttype=text/plain</code></td>
</tr>
</tbody>
</table>

12.2.5 **stream-open**

Creates a stream from an existing file.

Signature

```
stream-open(name:string, contenttype:string=contenttype=application/octet-stream) -> stream
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>The path of the source file for this stream.</td>
</tr>
<tr>
<td>contenttype</td>
<td>string</td>
<td>Specifies the <strong>contenttype</strong> to associate to the stream. The default is <code>contenttype=application/octet-stream</code></td>
</tr>
</tbody>
</table>

Examples

The following job opens an existing file having the `.txt` extension and writes it back to the same directory with the `.csv` extension:
Execution Steps

- **Execute function**: `/system/filesystem/copy`
- **Parameters**:
  - **Source**: `as-file(stream-open('C:\FlowForce\file.txt'))`
  - **Target**: `C:\FlowForce\file.csv`
  - **Overwrite**: ✔
  - **Abort on error**: ✔
  - **Working directory**: ✔
12.3 File System Functions

File system functions permit access to the file system. To execute these functions, the job must use the credentials of a user account with corresponding access rights on the operating system.

12.3.1 list-files

Lists the file/s specified by the path, which may end with a wildcard. It returns the string list. If the path does not end with a path separator and is not a wildcard, a search is made for exactly the specified item in the parent directory.

Signature

\[ \text{list-files(path:string)} \rightarrow \text{list of string} \]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>string</td>
<td>Specifies the path to a directory or file.</td>
</tr>
</tbody>
</table>

Examples

See Copy Files for an example.

12.3.2 list-directories

Lists the subdirectories in the path (which may terminate with a wildcard) and returns the resulting string list.

Signature

\[ \text{list-directories(path:string)} \rightarrow \text{list of string} \]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>string</td>
<td>Specifies the path to a directory.</td>
</tr>
</tbody>
</table>
12.3.3  **read-lines**

Reads the lines from the given file and returns them as a list of strings. The returned strings include the line ends (such as \n). You may need to trim each line with the help of the `trim()` function before processing it further, as illustrated in the example below.

**Signature**

```plaintext
read-lines(filename:string, encoding:string="UTF-8") -> list of string
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>string</td>
<td>Specifies the path to a file.</td>
</tr>
<tr>
<td>encoding</td>
<td>string</td>
<td>Specifies the encoding to use. The default encoding is ‘UTF-8’.</td>
</tr>
</tbody>
</table>

**Examples**

Let's suppose that you need to process multiple files that reside in multiple directories on the computer where FlowForce Server is installed. All the directory paths are saved as a text file, where each line corresponds to a directory path, for example:

```plaintext
C:\FlowForce\Examples\ListDirectories\1
C:\FlowForce\Examples\ListDirectories\2
C:\FlowForce\Examples\ListDirectories\3
```

The job illustrated below consists of two steps. The first step calls the `read-files` function and collects all directory paths from the text file above into a list. The second step iterates through the list of paths and calls the `list-files` function for each item. Note that the path is also trimmed before processing, to ensure that none of the resulting strings contain spaces or new line characters.
If you expose this job as a Web service and access it at the default address and port from a browser, the browser outputs the contents of each directory, as a JSON array, for example:

```json
0:
  0: "C:\FlowForce\Examples\ListDirectories\1\A.txt"
  1: "C:\FlowForce\Examples\ListDirectories\1\B.txt"
1:
  0: "C:\FlowForce\Examples\ListDirectories\2\C.txt"
  1: "C:\FlowForce\Examples\ListDirectories\2\D.txt"
2:
  0: "C:\FlowForce\Examples\ListDirectories\3\E.txt"
  1: "C:\FlowForce\Examples\ListDirectories\3\F.txt"
```
12.4 File Path Functions

File path functions allow you to extract specific portions of paths and file names. You may need to do this, for example, if you are polling a directory and want to extract the file name that triggered the job from the triggerfile parameter (see also File System Triggers).

12.4.1 extension

Extracts the file extension from a path.

Signature

```plaintext
extension(path:string) -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>string</td>
<td>Specifies the path to a file.</td>
</tr>
</tbody>
</table>

Examples

The following expression returns ".txt":

```plaintext
extension("c:\temp\file.txt")
```

12.4.2 filename

Extracts the file name (without extension) from a path.

Signature

```plaintext
filename(path:string) -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>string</td>
<td>Specifies the path to a file.</td>
</tr>
</tbody>
</table>
Examples
The following expression returns "file":

```
filename("c:\temp\file.txt")
```

### 12.4.3 filename-with-extension

Extracts the file name and extension from a path.

#### Signature

```
filename-with-extension(path:string) -> string
```

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>string</td>
<td>Specifies the path to a file.</td>
</tr>
</tbody>
</table>

#### Examples

The following expression returns "file.txt":

```
filename-with-extension("c:\temp\file.txt")
```

### 12.4.4 join-paths

Combines paths supplied as arguments into one path.

#### Signature

```
join-paths(string1:string, string2:string, stringN:string) -> string
```
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>string</td>
<td>Specifies a single path step to join. All subsequent arguments must be separated by a comma.</td>
</tr>
<tr>
<td>string2</td>
<td>string</td>
<td>Same as above.</td>
</tr>
<tr>
<td>stringN</td>
<td>string</td>
<td>Same as above.</td>
</tr>
</tbody>
</table>

Examples

On Windows, the following expressions return "C:\tmp\test.txt":

```plaintext
join-paths('C:\tmp', 'test.txt')
join-paths('C:\\', 'test.txt')
join-paths('C:\', 'tmp', 'test.txt')
join-paths('C:\Users', '\tmp', 'test.txt')
join-paths('D:\Data', 'C:\tmp', 'test.txt')
```

On Linux and MacOS, the following expressions return "/home/user/test.txt":

```plaintext
join-paths('/home/user', 'test.txt')
join-paths('/var', '/home/user', 'test.txt')
```

12.4.5 parent-directory

Extracts the parent directory from a path.

Signature

```plaintext
parent-directory(path: string) -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>string</td>
<td>Specifies the path to a directory.</td>
</tr>
</tbody>
</table>
Examples

Let's assume you have a MapForce mapping which updates a database from an XML file. You've deployed it to FlowForce Server already and created a job from it. Also, you've configured the job run when the content of the directory changes (that is, your job uses a file system trigger, see File System Triggers).

The first step of the job runs the mapping which updates the database:

After the mapping step finishes executing, your goal is to move the source XML file into the subdirectory called "processed". This would help you keep a track of which files have been processed. To achieve this goal, add a new step which calls the \texttt{/system/filesystem/move} function and enter as \texttt{Source} and \texttt{Destination} the values shown below:

The parameter value \texttt{(triggerfile)} in the \texttt{Source} field instructs FlowForce to move specifically the file which triggered the mapping. The parameter value \texttt{(parent-directory(triggerfile))} in the \texttt{Destination} field sets as destination a directory called "processed", inside the current directory. It consists of an expression and of a string. Note that only the expression part is delimited by curly braces (see Embedding Expressions in String Fields). The expression \texttt{(parent-directory(triggerfile))} calls the \texttt{parent-directory} function and supplies to it the value "triggerfile" as argument.

Therefore, when the job runs, the following actions take place:

1. A script or a user copies a file (let's call it \texttt{source.xml}) into the current working directory (for example, \texttt{C:FFSERV}).
2. The trigger fires and \texttt{source.xml} becomes the "triggerfile".
3. FlowForce Server executes the step which runs the mapping.
4. FlowForce Server executes the step which moves `source.xml` to the "processed" subdirectory. Note that the path `C:\FFSERV\processed` must exist.
12.5  List Functions

List functions are used to create and disassemble lists. Lists always contain items of a single type (for example, only strings, only number, or only nested lists with the same item type); there are no mixed type lists.

12.5.1  char

Returns a string that contains the Unicode character of the number supplied as argument. For example, char(10) returns a Line Feed. To find out the numeric code of a specific Unicode character, use the code function.

Signature

\[ \text{char(number: number) \rightarrow string} \]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>number</td>
<td>The numeric code of the character. This code is equivalent to the decimal code used to represent a Unicode character in HTML (for example, 8734 represents the infinity symbol).</td>
</tr>
</tbody>
</table>

Examples

The following execution step returns the infinity symbol:
12.5.2  code

Returns the Unicode value of the first character of the string supplied as argument.

Signature

code(string: string) -> number

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>Specifies the input string.</td>
</tr>
</tbody>
</table>

Examples

The following execution step returns the numeric value 32, which represents the space character:
### 12.5.3 from-to

Returns the list of integers between "from" and "to" inclusive. If "from" is greater than "to", this list is empty.

**Signature**

```
from-to(from: number, to: number) -> list of number
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>number</td>
<td>Specifies the starting index (&quot;from&quot;).</td>
</tr>
<tr>
<td>to</td>
<td>number</td>
<td>Specifies the ending index (&quot;to&quot;).</td>
</tr>
</tbody>
</table>

**Examples**

The following expression produces \([3, 4, 5, 6, 7]\):

```
from-to(3, 7)
```

### 12.5.4 join

Concatenates the lists given by the first argument using the second argument as separator between each pair of lists.
Signature

\[
\text{join} (\text{lists: list of lists}, \text{separator: list=empty list}) \rightarrow \text{list}
\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>list of lists</td>
<td>Specifies the lists to join. This argument must be a list of two or more lists. All nested lists must be of the same type.</td>
</tr>
<tr>
<td>separator</td>
<td>list</td>
<td>Optional argument which specifies the separator by which to delimit the joined lists. If not supplied, no separator will be used. The separator must be of type list. Use the list function to create a separator. For example, the expression list(’,’) specifies a single comma character as separator.</td>
</tr>
</tbody>
</table>

Examples

The following execution steps illustrate how to join two lists. Step 1 produces the first list. Step 2 produces the second list. Step 3 creates an object of type "list of lists" that contains both lists. Finally, step 4 joins the lists, using the semi-colon character as separator.
### 12.5.5 length

Returns the number of items in the list.

#### Signature

```
length(list: list) -> number
```

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>list</td>
<td>Specifies the input list object.</td>
</tr>
</tbody>
</table>
12.5.6  **list**

Builds a list from single items. All items must be of the same type, the resulting list is a list of items of that type.

**Signature**

```plaintext
list(item1: any type, item2: any type, itemN: any type) -> list
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item1</td>
<td>any type</td>
<td>Specifies a single item. Subsequent items must be separated by a comma.</td>
</tr>
<tr>
<td>item2</td>
<td>any type</td>
<td>Same as above</td>
</tr>
<tr>
<td>itemN</td>
<td>any type</td>
<td>Same as above.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression returns the list `[1, 2, 3]`. All list items are of numeric type:

```plaintext
list(1,2,3)
```

The following expression returns the list `['a', 'b', 'c']`. All list items are of string type:

```plaintext
list('a','b','c')
```

12.5.7  **nth**

Returns the specified item from the list. The index is zero-based. Fails if the index is out of bounds.

**Signature**

```plaintext
nth(list: list, index: number) -> item
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>list</td>
<td>Specifies the input list.</td>
</tr>
</tbody>
</table>
### List Functions

#### Expression Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>number</td>
<td>Specifies the zero-based index of the item to return.</td>
</tr>
</tbody>
</table>

#### Examples

The following expression returns "b":

\[
\text{nth(list('a', 'b', 'c'), 1)}
\]

### 12.5.8 slice

Returns a partial list from a list.

**Signature**

\[
\text{slice(list: list, start: number, end: number} = \text{length(list)} \rightarrow \text{list}
\]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>list</td>
<td>Specifies the input list.</td>
</tr>
<tr>
<td>start</td>
<td>number</td>
<td>Specifies the zero-based index of the first list item to include in the slice.</td>
</tr>
<tr>
<td>end</td>
<td>number</td>
<td>Specifies the zero-based index of the first item to ignore in the slice.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression returns `list(2,3)`:

\[
\text{slice(list(1,2,3,4),1,3)}
\]
12.6 String Functions

The string functions perform basic string operations, such as concatenation, extracting a substring from a string, trimming, splitting, and others.

12.6.1 concat

Concatenates the strings supplied as arguments into one string. To concatenate all items of an object of type "list of string", use the string-join function.

Signature

```
concat(string1:string, string2:string, stringN:string) -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>string</td>
<td>Specifies a single string item to join. All subsequent arguments must be separated by a comma.</td>
</tr>
<tr>
<td>string2</td>
<td>string</td>
<td>Same as above.</td>
</tr>
<tr>
<td>stringN</td>
<td>string</td>
<td>Same as above.</td>
</tr>
</tbody>
</table>

Examples

The following expression returns "abc":

```
concat('a', 'b', 'c')
```

12.6.2 contains

Returns true if the first string contains at least one occurrence of substring, otherwise false.

Signature

```
contains(string:string, substring:string) -> Boolean
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
</tbody>
</table>
### String Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>substring</td>
<td>string</td>
<td>The string value to check for.</td>
</tr>
</tbody>
</table>

#### Examples

The following expression returns **true**:

```plaintext
contains('cat','a')
```

The following expression returns **false**:

```plaintext
contains('cat','b')
```

#### 12.6.3  `ends-with`

Returns **true** if the string supplied in the `string` argument ends with the string supplied in the `end` argument.

**Signature**

```plaintext
ends-with(string:string, end:string) -> Boolean
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
<tr>
<td>end</td>
<td>string</td>
<td>The string value to check for.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression returns **true**:

```plaintext
ends-with('cat', 't')
```

The following expression returns **false**:

```plaintext
ends-with('cat', 'a')
```
### 12.6.4 find-all

Extracts all occurrences of \texttt{pattern} in the string, where \texttt{pattern} is a regular expression.

**Signature**
\[
\text{find-all} (\text{string: string, pattern: string}) \rightarrow \text{list of string}
\]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
<tr>
<td>pattern</td>
<td>string</td>
<td>The pattern as a regular expression.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression extracts all occurrences of "o" from string "apollo".

\[
\text{find-all('apollo', 'o')}
\]

The result is the following list of string: ["o", "o"]

### 12.6.5 number

Computes the number representation of the string, i.e. converts the string supplied as argument into a number.

**Signature**
\[
\text{number} (\text{string: string}) \rightarrow \text{number}
\]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string value to convert.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression converts the string value "1" into the numeric value 1:

\[
\text{number('1')}
\]
12.6.6 split

Splits the string supplied as argument at each occurrence of separator.

**Signature**

`split(string: string, separator: string) -> list of string`

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
<tr>
<td>separator</td>
<td>string</td>
<td>The separator string.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression will return the list ['1', '2', '3']:

```plaintext```
split('1;2;3', ';')
```plaintext```

12.6.7 starts-with

Returns true if the string supplied in the string argument starts with the string supplied in the start argument.

**Signature**

`starts-with(string: string, start: string) -> Boolean`

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
<tr>
<td>start</td>
<td>string</td>
<td>The string value to check for.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression returns true:

```plaintext```
starts-with('cat', 'c')
```plaintext```

The following expression returns false:
12.6.8  string

Computes the string representation of the given number, i.e. converts the number supplied as argument into a string.

Signature

```
string(number: number) -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>number</td>
<td>The number to be convert to string.</td>
</tr>
</tbody>
</table>

Examples

The following expression converts the numeric value 1 into the string "1":

```
string(1)
```

12.6.9  string-join

Joins the list of strings supplied as argument into a string. Optionally, inserts the separator supplied as argument in between each string.

Signature

```
string-join(list:list of string, separator:string="") -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>list of string</td>
<td>The input list of string.</td>
</tr>
<tr>
<td>separator</td>
<td>string</td>
<td>Optional argument. Specifies the separator by which all joined strings should be delimited.</td>
</tr>
</tbody>
</table>
Examples
The following expression will return the string a;b;c:

\[\text{string-join(list('a', 'b', 'c'), ';')}\]

12.6.10 string-length

Returns the number of characters in the string.

Signature

\[\text{string-length(string: string) \rightarrow number}\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
</tbody>
</table>

Examples
The following expression will return 3:

\[\text{string-length('cat')}\]

12.6.11 substring

Returns a substring from the specified string, beginning with start character position, up to the end character position. The start and end indexes are zero-based.

If not set, end is the length of the supplied string.

The end argument can also be a negative integer. A negative value -n means "trim the last n characters from the string".

Signature

\[\text{substring(string: string, start: number, end: number) \rightarrow string}\]
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
<tr>
<td>start</td>
<td>number</td>
<td>The zero-based starting index.</td>
</tr>
<tr>
<td>end</td>
<td>number</td>
<td>The zero-based ending index.</td>
</tr>
</tbody>
</table>

Examples

The following expression will return "Force":

```
substr('FlowForce',4)
```

The following expression will return "t":

```
substr('Altova',2,3)
```

The following expression will return "Itov":

```
substr('Altova',1,-1)
```

### 12.6.12 trim

Removes leading and trailing whitespace characters from the string (Space, Tab, Line Feed, Carriage Return, Form Feed, and Vertical Tab).

Signature

```
trim(string:string) -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
</tbody>
</table>
12.6.13 trim-start

Removes leading whitespace from the string supplied as argument (see also the trim function).

Signature

\[
\text{trim-start}(\text{string}: \text{string}) \rightarrow \text{string}
\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
</tbody>
</table>

12.6.14 trim-end

Removes trailing whitespace from the string supplied as argument (see also the trim function).

Signature

\[
\text{trim-end}(\text{string}: \text{string}) \rightarrow \text{string}
\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
</tbody>
</table>
12.7 Boolean Functions

The Boolean functions are used to evaluate true/false expressions.

12.7.1 all

Returns true if all Boolean values are true; stops evaluation after the first false value and returns false.

Signature

all(booVal1:Boolean, boolVal2:Boolean, boolValN:Boolean) → Boolean

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>booVal1</td>
<td>Boolean</td>
<td>Specifies a Boolean value to evaluate. Subsequent values must be separated by a comma.</td>
</tr>
<tr>
<td>boolVal2</td>
<td>Boolean</td>
<td>Same as above.</td>
</tr>
<tr>
<td>boolValN</td>
<td>Boolean</td>
<td>Same as above.</td>
</tr>
</tbody>
</table>

12.7.2 any

Returns true if any Boolean value is true; stops evaluation after the first true value. Returns false if all values are false.

Signature

any(boolVal1:Boolean, boolVal2:Boolean, boolValN:Boolean) → Boolean

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolVal1</td>
<td>Boolean</td>
<td>Specifies a Boolean value to evaluate. Subsequent values must be separated by a comma.</td>
</tr>
<tr>
<td>boolVal2</td>
<td>Boolean</td>
<td>Same as above.</td>
</tr>
<tr>
<td>boolValN</td>
<td>Boolean</td>
<td>Same as above.</td>
</tr>
</tbody>
</table>
12.7.3  **false**

Returns Boolean **false**.

**Signature**

```
false() -> Boolean
```

12.7.4  **if**

Returns **valueTrue** if the Boolean condition is true, and **valueFalse** if false. Only the selected subexpression is evaluated. Both subexpressions must be of the same type, which is also the return type.

**Signature**

```
if(condition: Boolean, valueTrue: any type, valueFalse: any type) -> any type
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>Boolean</td>
<td>Specifies the condition to evaluate.</td>
</tr>
<tr>
<td>valueTrue</td>
<td>any type</td>
<td>Specifies a subexpression to return when <strong>condition</strong> evaluates to <strong>true</strong>.</td>
</tr>
<tr>
<td>valueFalse</td>
<td>any type</td>
<td>Specifies a subexpression to return when <strong>condition</strong> evaluates to <strong>false</strong>.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression passes a Boolean as XML Schema conformant value:

```
if(b, "true", "false")
```

An alternative way to do this:

```
if(b, "1", "0")
```
12.7.5  not

Returns the negation of the Boolean value supplied as argument.

**Signature**

```
not(value: Boolean) -> Boolean
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Boolean</td>
<td>Specifies the Boolean value to negate.</td>
</tr>
</tbody>
</table>

12.7.6  true

Returns Boolean `true`.

**Signature**

```
true() -> Boolean
```
12.8  Runtime Information Functions

The runtime information functions can be used to handle the details of the currently running jobs.

12.8.1  instance-id

Returns a unique string for every job execution. This can be used to create a unique directory for each job execution, where the string is used to define the directory name.

Signature

\[
\text{instance-id}() \rightarrow \text{string}
\]

12.8.2  log

Converts the expression received as argument to string and writes it to the system log. This function is useful in situations where you want to explicitly log the expression produced by a step. Logging values this way has the effect that no truncation of values occurs in the system log when the logged values are too long, see also Logging Settings.

Signature

\[
\text{log}(\text{expression}: T_0) \rightarrow \text{string}
\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>T_0</td>
<td>The FlowForce expression to be logged, of type T_0 (any type).</td>
</tr>
</tbody>
</table>

Examples

Let's assume that you have created a job which gets a list of files from the given path, like the one below.

![Execution Steps]

- Execute function `/system/compute`
  - Parameters: Expression: `list-files('C:\FlowForceExamples\LogFunction\source')`
  - Assign this step's result to `name` as T_0
If the number of files in the source directory exceeds the FlowForce default logging limit for lists, then entries in the job log become truncated. As illustrated below, in this example, only the first 10 file names are shown. Also, the last character in each file path has been truncated, because the path has exceeded the default limit of 50 characters.

To prevent truncation from happening, enclose the expression inside the log function, and save the job configuration.

If you run the job with the new configuration, the log now contains a new entry for the logged expression, in addition to the entry logged by the system. Truncation no longer occurs.
In this example, as an alternative to calling the log expression function, you can also click the "Enable logging" button next to the step parameter you wish to log. Doing this is equivalent to using the log function, so FlowForce will hide the log function next time when you open the job configuration page. The difference between the button and the log function is that the former logs the entire expression displayed in the text box, whereas the log function can be used selectively for smaller sub-expressions, for example:

```
12.8.3 slot-number

Returns the execution slot number of the queue currently running the job. This number should not be used as a file name. The number can be used to access different servers to execute parallel jobs (simple load balancing).

The slot number depends on the queue in which the slot execution was started. If the current job is called by another job, then it inherits the slot number of the calling job.

Signature

slot-number() -> number
```
12.9 MIME Expression Functions

The MIME (Multipurpose Internet Mail Extensions) message format specifies what type of content a message has and how this message is encoded. To find out more about MIME, see the Microsoft documentation. You can use MIME expression functions when you need to manipulate MIME headers.

The following MIME expression functions are available:

- `add-mime-header`
- `add-mime-headers`
- `current-message-id`
- `get-mime-content-disposition-param`
- `get-mime-content-id`
- `get-mime-content-type-param`
- `get-mime-header`
- `get-mime-headers`
- `get-stream-filename`
- `is-file`
- `is-mime-content-type`
- `mime-content-encode`
- `mime-flatten`
- `mime-multipart`
- `mime-multipart-related`
- `mime-multipart-from-list`
- `mime-parse`
- `mime-split-multipart`
- `new-message-id`
- `reset-mime-headers`
- `set-mime-content-disposition`
- `set-mime-content-id`
- `set-mime-header`
- `set-mime-headers`

12.9.1 add-mime-header

Returns a stream with added header `key: value`. This function does not remove an existing header with that key.

Signature

```
add-mime-header(s: stream, key: string, value: string) -> stream
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td><code>stream</code></td>
<td>The stream to which the header should be added.</td>
</tr>
</tbody>
</table>
### 12.9.2 add-mime-headers

Returns a stream with all headers from `headers` added.

**Signature**

```
add-mime-headers(s:stream, headers:list of (string, string)) -> stream
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>headers</td>
<td>list of (string, string)</td>
<td>The list of headers to be added. Use the list function to create a list.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression returns a stream with two headers: **Content-Disposition**, and **Content-Transfer-Encoding**.

```
add-mime-headers(empty-stream(), list(('Content-Disposition','attachment; name=something'), ('Content-Transfer-Encoding','7bit')))```

### 12.9.3 current-message-id

Returns the **Message-ID** header field of an AS2 message. This function must be used in a job that is configured to receive AS2 requests. That is, the check box **Make this job available via HTTP at URL...** must
be selected in the job configuration page. Otherwise, this function returns a newly generated Message-ID (a new value is generated whenever a new job instance runs and stays constant for that job instance until it ends).

**Signature**

```
current-message-id() -> string
```

**Examples**

The following expression produces a filename based on the **Message-ID**. The `substring` function removes the angle brackets (the first and last character) from the **Message-ID**.

```
C:\temp\{substring(current-message-id(), 1, -1)}.msg
```

The following expression does the same as above, and additionally splits the current **Message-ID** apart at character '@' with the help of the `split` function. The `nth` function extracts only the first part—a random hexadecimal value 32 characters long—and uses that as part of a filename.

```
C:\temp\{nth(split(substring(current-message-id(), 1, -1), '@'), 0)}.msg
```

### 12.9.4 `get-mime-content-disposition-param`

Returns the parameter `param` from the "Content-Disposition" header of a stream if such header and parameter exists; otherwise, it returns the value of the `default` argument. This function can be used to receive messages that follow the optional AS2 profile **FileName preservation (FN)** to extract the original file name from the MIME header.

**Signature**

```
get-mime-content-disposition-param(s:stream, param:string, default:string="") -> string
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>param</td>
<td>string</td>
<td>Specifies the name of the parameter to return.</td>
</tr>
<tr>
<td>default</td>
<td>string</td>
<td>Specifies the value to return when the specified <code>param</code> and header do not exist. By default, this is an empty string.</td>
</tr>
</tbody>
</table>
Examples
Assuming that stream \texttt{msg} contains the header \texttt{Content-Disposition: attachment; filename="GETMSG.edi"}, the following expression will return "GETMSG.edi":

\begin{verbatim}
get-mime-content-disposition-param(msg, "filename")
\end{verbatim}

12.9.5 \textbf{get-mime-content-id}

Returns the value of the \texttt{Content-ID} header from the stream supplied as argument, if such header exists; otherwise, it returns the value of the \texttt{default} argument.

\textbf{Signature}

\begin{verbatim}
get-mime-content-id(s:stream, default:string="") -> string
\end{verbatim}

\textbf{Parameters}

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>default</td>
<td>string</td>
<td>Specifies the value to return when \texttt{Content-ID} header does not exist. By default, this is an empty string.</td>
</tr>
</tbody>
</table>

\textbf{Examples}

Let's suppose that stream \texttt{msg} has the header \texttt{Content-ID: <root.attachment>}. The expression

\begin{verbatim}
get-mime-content-id(msg, "")
\end{verbatim}

returns "<root.attachment>" in this case. If no such header exists, the expression above returns an empty string (the value of the second argument).

12.9.6 \textbf{get-mime-content-type-param}

Returns the parameter \texttt{param} from the "Content-Type" header of a stream if such header and parameter exists; otherwise, it returns the value of the \texttt{default} argument. This function can be used to receive messages that follow the optional AS2 profile \textbf{Multiple Attachments (MA)}. Namely, it can extract the starting document \texttt{Content-ID} and \texttt{Content-Type} specified as parameters 'start' and 'type' to \texttt{multipart/related} content type. It can also be used to extract the character set, as shown in the example below.
Signature

```plaintext
get-mime-content-type-param(s:stream, param:string, default:string="") -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>param</td>
<td>string</td>
<td>Specifies the name of the parameter to return.</td>
</tr>
<tr>
<td>default</td>
<td>string</td>
<td>Specifies the value to return when the requested <code>param</code> does not exist. By default, this is an empty string.</td>
</tr>
</tbody>
</table>

Examples

Assuming that stream `msg` contains the header `Content-Type: text/html; charset=utf-8`, the following expression will return "utf-8":

```plaintext
get-mime-content-type-param(msg, "charset", "ascii")
```

12.9.7 `get-mime-header`

Gets a specific MIME header from the current stream if such a header exists; otherwise, it returns the value of the `default` argument.

Signature

```plaintext
get-mime-header(s:stream, key:string, default:string="") -> string
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>key</td>
<td>string</td>
<td>The <code>key</code> from the key-value pair that forms the header.</td>
</tr>
<tr>
<td>default</td>
<td>string</td>
<td>Specifies the default value to return. By default, this is an empty string.</td>
</tr>
</tbody>
</table>
Examples
Assuming that stream \texttt{msg} contains the header \texttt{Content-Disposition: attachment; filename="GETMSG.edi"}, the following expression will return \texttt{attachment; filename="GETMSG.edi":}

\begin{verbatim}
get-mime-header(msg, "Content-Disposition", "")
\end{verbatim}

In this example, if the stream does not have the "Content-Disposition" header, the expression above will return an empty string (the value of the third argument).

\textbf{12.9.8 get-mime-headers}

Gets all MIME headers from a stream and returns a list of tuples (key, value). The returned list can be supplied as \texttt{headers} parameter to the \texttt{add-mime-headers} expression function.

\underline{Signature}

\begin{verbatim}
get-mime-headers(s:stream) \rightarrow list of (string, string)
\end{verbatim}

\underline{Parameters}

\begin{tabular}{|l|l|l|}
\hline
Name & Type & Description \\
\hline
\texttt{s} & \texttt{stream} & Specifies the input stream. \\
\hline
\end{tabular}

\textbf{12.9.9 get-stream-filename}

Returns a stream's file name with extension if the stream supplied as argument was created from a file. Otherwise, it returns the value of the \texttt{default} argument.

\underline{Signature}

\begin{verbatim}
get-stream-filename(stream:stream, default:string="") \rightarrow string
\end{verbatim}

\underline{Parameters}

\begin{tabular}{|l|l|l|}
\hline
Name & Type & Description \\
\hline
\texttt{stream} & \texttt{stream} & Specifies the input stream. \\
\texttt{default} & \texttt{string} & Specifies the default value to return. By default, this is an empty string. \\
\hline
\end{tabular}
12.9.10  is-file

Returns **true** if the function `as-file` would return the name of an existing file, and **false** if `as-file` would create a temporary file.

For example, it returns **true** if the stream was created from a file using the `stream-open` function or returned from a mapping. If the stream is not served from a file or it is a file but a temporary one, this function returns **false**.

**Signature**

```plaintext
is-file(s:stream) -> Boolean
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
</tbody>
</table>

12.9.11  is-mime-content-type

Matches the "Content-Type" header of the stream to custom-defined accept rules. Returns **true** if the "Content-Type" header exists and the rules match its value, otherwise returns **false**. A stream without "Content-Type" header will be treated as "application/octet-stream".

The accept rules have the following format, in extended Backus-Naur form (EBNF) notation:

```plaintext
Match ::= Single ("," Single)*
Single ::= Spaces? Type-Match (Spaces? ";" Spaces? Parameter )* Spaces?
Type-Match ::= 
   
Type "/+" Suffix |
Type "/" Subtype |

Parameter ::= Name "=" Value
```

**Signature**

```plaintext
is-mime-content-type(s:stream, accept:string) -> Boolean
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
</tbody>
</table>
### Expression Functions

#### 12.9.12 mime-content-encode

Applies encoding as Content-Transfer-Encoding to stream \( s \).

The supported encodings are:

- Empty string: Equivalent to "binary".
- "base64": Base64 encoding
- "quoted-printable": Quoted printable encoding
- Any other string: No encoding

The function decodes the stream using the current Content-Transfer-Encoding and re-encodes it using the specified encoding. The new Content-Transfer-Encoding is stored in the headers of the resulting stream.

The function does not guarantee that errors in the source encoding are reported.

**Signature**

\[
\text{mime-content-encode}(s: \text{stream}, \text{encoding}: \text{string}="") \rightarrow \text{stream}
\]
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>encoding</td>
<td>string</td>
<td>Specifies the encoding to apply. By default, this is an empty string.</td>
</tr>
</tbody>
</table>

### 12.9.13 mime-flatten

Takes a stream with MIME headers and converts it to a stream that includes the original headers in the content. The resulting stream will have a content type of "message/rfc822".

**Signature**

```
mime-flatten(s:stream) -> stream
```  

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
</tbody>
</table>

### 12.9.14 mime-multipart

Takes any number of streams and combines them into a multipart/subtype.

The boundary is invented automatically. The streams will be flattened before assembly. Multipart with additional parameters are not yet supported.

**Note for FlowForce Server Advanced Edition users:** The subtype should always be related for AS2, as AS2 does not define a meaning for other multipart messages. See also the `mime-multipart-related` function.

**Signature**

```
mime-multipart(subtype:string, s:stream) -> stream
```  

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subtype</td>
<td>string</td>
<td>Specifies the multipart/subtype to use.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
</tbody>
</table>

### Examples

The following expression returns a stream that includes two files, an EDI file and a PDF.

```plaintext
mime-multipart("related", stream-open("c:\example\order.edi", "application/EDIFACT"), stream-open("c:\example\measuredetails.pdf", "application/pdf"))
```

### Execution Steps

![Execution Steps](image)

### 12.9.15 mime-multipart-related

Takes any number of streams and combines them into a multipart/related. The boundary is invented automatically. The streams will be flattened before assembly.

**Note for FlowForce Server Advanced Edition users:** This function can be used to assemble a message that follows the optional AS2 profile Multiple Attachments (MA). The first stream will become a main part. All the parts get the "Content-ID" header with invented unique values before assembling multipart, if they don't have it. The invented value is a new **Message-ID** as returned by the `new-message-id` function. Source streams are not affected.

**Signature**

```
mime-multipart-related(s:list of stream) -> stream
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>list of stream</td>
<td>Specifies the input list of streams.</td>
</tr>
</tbody>
</table>

### Examples

The following expression returns a stream that includes two streams.

```plaintext
mime-multipart-related(list(part1, part2))
```
12.9.16  mime-multipart-from-list

Takes a list of streams and combines them into a multipart/subtype.

Signature

```
mime-multipart-from-list(subtype:string, s:list of stream) -> stream
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subtype</td>
<td>string</td>
<td>Specifies the multipart/subtype to use.</td>
</tr>
<tr>
<td>s</td>
<td>list of stream</td>
<td>Specifies the input list of streams.</td>
</tr>
</tbody>
</table>

12.9.17  mime-parse

Parses a MIME message stored in stream s, and separates MIME headers and message body. Returns a stream that has message body content, decoded according to the "Content-Transfer-Encoding" header if needed. MIME headers are accessible via expression functions, like get-mime-header, is-mime-content-type and such. Reverts what was done by mime-flatten function. The function does not guarantee that errors in the source stream are reported.
12.9.18 mime-split-multipart

If stream $s$ is a MIME multipart message, this function splits it and return a list of streams. If stream $s$ is not a multipart message (that is, if `is-mime-content-type(s, "multipart/*")` returns `false`), then the function returns a list of one element—stream $s$ (unchanged). The function does not guarantee that errors in the source stream are reported.

12.9.19 new-message-id

Generates and returns a new value for the **Message-ID** header field. You can use this value to populate the header of a MIME message. This function, unlike `current-message-id`, always returns a new **Message-ID**. The **Message-ID** has the following format:

'<' UTC timestamp '-' random hex value 32 characters long '@' host name related text '>'

For example: `<20180306154822808383-5933b654b26c4495bb0b619ab72b3bc60myservename>`. 
12.9.20  reset-mime-headers

Returns a stream with completely fresh headers. Without a header list, it clears all headers.

Signature

reset-mime-headers(s: stream, headers: list of (string, string)=empty) -> stream

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>headers</td>
<td>list of (string, string)</td>
<td>Specifies the list of headers to create. The default value is empty.</td>
</tr>
</tbody>
</table>

12.9.21  set-mime-content-disposition

Sets the parameter of a MIME Content-Disposition header found in stream s.

*FlowForce Server Advanced Edition:* This function is useful when you send AS2 messages with the optional AS2 profile FileName preservation (FN). See also the get-mime-content-disposition-param function for reading the file name.

Signature

set-mime-content-disposition(s: stream, disposition: string, filename: string="") -> string

Parameters

- s (type: stream) specifies an input stream.
- disposition (type: string) specifies the disposition value of the Content-Disposition header.
- filename (type: string) specifies the filename value of the Content-Disposition header. By default, this is an empty string.

*Example*

The following expression sets the Content-Disposition header as follows: set-mime-content-disposition(msg, "attachment", "GETMSG.edi"). You can use the set-mime-content-disposition function to make a file downloadable.
12.9.22 set-mime-content-id

Returns a stream with the "Content-ID" header set to \textit{value}, and all other headers and content untouched. You can also achieve the same result using the \textit{set-mime-header} function; this function represents a more direct approach.

\textbf{Signature}

\begin{verbatim}
set-mime-content-id(s:stream, value:string="") -> stream
\end{verbatim}

\textbf{Parameters}

\begin{tabular}{|l|l|l|}
\hline
\textbf{Name} & \textbf{Type} & \textbf{Description} \\
\hline
s & stream & Specifies the input string. \\
\hline
value & string & Specifies the value to set in the "Content-Disposition". \\
\hline
\end{tabular}

\textbf{Examples}

Let's assume that you want to set the value of the "Content-ID" header in stream \textit{msg} to \texttt{<root.attachment>}. To do this, use the following expression:

\begin{verbatim}
set-mime-content-id(msg, "<root.attachment>")
\end{verbatim}

12.9.23 set-mime-header

Returns a stream with header \textit{key} set to \textit{value}, and all other headers and content untouched. If you need to change several headers at once, you might want to use the \textit{set-mime-headers} function.

\textbf{Signature}

\begin{verbatim}
set-mime-header(s:stream, key:string, value:string) -> stream
\end{verbatim}

\textbf{Parameters}

\begin{tabular}{|l|l|l|}
\hline
\textbf{Name} & \textbf{Type} & \textbf{Description} \\
\hline
s & stream & Specifies the input stream. \\
\hline
key & string & Specifies the key of the header to set. \\
\hline
value & string & Specifies the header value to set. \\
\hline
\end{tabular}
Examples
To override the "Content-Type" header, use:

```plaintext
set-mime-header(s, "Content-Type", "text/plain; charset=iso-8859-1")
```

12.9.24 set-mime-headers

Returns a stream with headers augmented by the key-value pairs from `headers`. The new headers will replace any existing headers of the same name.

Signature

```plaintext
set-mime-headers(s:stream, headers:list of (string, string)) -> stream
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>stream</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>headers</td>
<td>list of (string, string)</td>
<td>The list of headers to set.</td>
</tr>
</tbody>
</table>

Examples
To override the "Content-Type" header, use:

```plaintext
set-mime-headers(s, list(("Content-Type", "text/plain; charset=iso-8859-1")))
```
## 13 Job Configuration Examples

This chapter includes step-by-step FlowForce job configuration examples. The table below lists all the examples, along with the specific function kinds and triggers illustrated in each example.

<table>
<thead>
<tr>
<th>Example</th>
<th>Concepts illustrated</th>
<th>Built-in functions</th>
<th>Expression functions</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a &quot;Hello, World!&quot; Job</td>
<td></td>
<td>• /system/compute</td>
<td></td>
<td>Web service</td>
</tr>
<tr>
<td>Check if a Path Exists</td>
<td></td>
<td>• /system/shell/commandline</td>
<td>• /system/compute-string</td>
<td>Web service</td>
</tr>
<tr>
<td>Copy Files</td>
<td></td>
<td>• /system/filesystem/copy</td>
<td></td>
<td>Web service</td>
</tr>
<tr>
<td>Create a Job from a MapForce Mapping</td>
<td>• MapForce mapping</td>
<td>• /system/filesystem/copy</td>
<td></td>
<td>Timer</td>
</tr>
<tr>
<td>Use a Job as Step of Another Job</td>
<td>• /system/filesystem/copy</td>
<td>• /system/shell/commandline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a Directory Polling Job</td>
<td>• MapForce mapping</td>
<td>• /system/filesystem/move</td>
<td></td>
<td>File system</td>
</tr>
<tr>
<td>Add Error Handling to a Job</td>
<td></td>
<td>• /system/shell/commandline</td>
<td>• /system/mail/send</td>
<td>Web service</td>
</tr>
<tr>
<td>Expose a Job as a Web Service</td>
<td></td>
<td>• MapForce mapping</td>
<td></td>
<td>Web service</td>
</tr>
<tr>
<td>Post JSON to FlowForce Web Service</td>
<td>• /system/filesystem/copy</td>
<td></td>
<td></td>
<td>Web service</td>
</tr>
<tr>
<td>Cache Job Results</td>
<td></td>
<td>• /system/shell/commandline</td>
<td></td>
<td>Web service</td>
</tr>
<tr>
<td>Create a Job from a StyleVision Transformation</td>
<td>• StyleVision transformation</td>
<td>• /system/compute</td>
<td></td>
<td>Timer</td>
</tr>
<tr>
<td>Validate a Document with RaptorXML</td>
<td>• /RaptorXML/valany</td>
<td>• /system/compute</td>
<td></td>
<td>Timer</td>
</tr>
<tr>
<td>Validate XML with Error Logging</td>
<td>• /RaptorXML/valxml-withxsd</td>
<td>• /system/compute</td>
<td></td>
<td>Web service</td>
</tr>
<tr>
<td>Run XSLT with RaptorXML</td>
<td>• /RaptorXML/xslt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Job Configuration Examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Concepts illustrated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in functions</td>
</tr>
<tr>
<td>Generate PDFs from XML Files</td>
<td>• MapForce mapping</td>
</tr>
<tr>
<td></td>
<td>• StyleVision transformation</td>
</tr>
<tr>
<td></td>
<td>• /system/compute</td>
</tr>
</tbody>
</table>
13.1 Create a "Hello, World!" Job

This example shows you how to create a simple job that outputs the text "Hello, World!" in the browser. The text will be created by means of a FlowForce expression. You will be able to trigger the job on demand by clicking a link in the browser (that is, the job will be exposed as a Web service).

Prerequisites

- Required licenses: FlowForce Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container used in this example is accessible to any authenticated user).

Creating the job

1. Log on to FlowForce Server and navigate to the /public container.
2. Click Create | Create Container and create a new container called "Examples".

   The /public/Examples container is used by convention in most of the jobs illustrated in this documentation. You can create your jobs in any other containers as well, but if you want to follow all the subsequent tutorials from this documentation literally, it is recommended to create the /public/Examples container.

3. In the /public/Examples container, click Create | Create job, and enter the job title and description.

4. Add a new execution step which calls the built-in function /system/compute.

5. In the Expression field, enter the text 'Hello, World', enclosed within single quotes. The content of this field represents a FlowForce Server expression.
6. Declare the execution result as **string**.

7. Select the **Make this job available via HTTP...** check box and type "HelloWorldService" as service name. For more information, see [Exposing Jobs as Web Services](#).

8. Under "Credentials", select an existing credential record or specify a local credential. For more information, see [Credentials](#).

9. Click **Save**.

### Running the job

You have now finished creating a job that computes the string value "Hello, World!" and returns it as the job result. To run the job, do one of the following:

- Go to **Home**, and then click **Show all active triggers and services**. Next, click the job's URL displayed in the "Info" column.
- Enter `http://127.0.0.1:4646/service/HelloWorldService` in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the **Configuration page**, change the address accordingly.
- If you set the optional **Host name** field of FlowForce Server from the **Setup Page**, you can execute the web service call directly from the job configuration page, by clicking the **Play** button adjacent to the **Make this job available via HTTP** check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

---

**Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the Service - Use permission to this user on the container where the job is, and then access the Web service with the corresponding**
Create a "Hello, World!" Job

user account. To disable HTTP authentication and make the Web service public, grant the Service - Use permission to the user Anonymous, see How Permissions Work.

If the job executes successfully, the browser displays the output of the job:

Hello, World!

If the job fails, the browser displays a "Service execution failed" message. In this case, check the FlowForce Server job log to identify the error.
13.2  Check if a Path Exists

This example shows you how to create a job which informs you if a path (to a file or directory) exists on the operating system. To achieve this goal, you will use a combination of built-in functions and expression functions. The job will be defined as a Web service, so that you can trigger it on demand, by accessing a URL from the browser. The job will take the path as an argument, and will return a string which informs whether the path supplied as argument exists on the operating system where FlowForce Server runs.

Prerequisites

- Required licenses: FlowForce Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port.
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container used in this example is accessible to any authenticated user).

Creating the job

1. Log on to FlowForce Server and navigate to a container where you have permission to create new jobs. For consistency with other examples, this tutorial uses the /public/Examples container—if you don't have this container yet, create it using the Create | Create Container command.
2. In the /public/Examples container, click Create, and then select Create job.
3. Add a job name ("CheckPath", in this example) and, optionally, a job description.
4. Under Job Input Parameters, click +, and add the parameter path, as shown below.
5. Add a new execution step which calls the /system/shell/commandline function, and enter the shell command which checks for the existence of the file. Make sure to declare the result of this step, as shown below (in this example, we called it output).
On Windows, the shell command outputs "1" when the path exists and "0" when it does not exist. If FlowForce Server runs on a Unix system, adjust the command accordingly. Notice that the command embeds the FlowForce expression \{path\}. This expression references the input parameter defined in the previous step.

6. Under "Execution Steps", click the button, and then select new Choose step. Then enter \texttt{trim(content(stdout(output))) == '1'} as condition expression. This expression consists of three nested functions: stdout, content, and trim. First, the stdout function gets the standard output of the result returned by the previous step. Then the content function converts the standard output to string. Finally, the trim function removes any leading or trailing spaces, carriage returns, or line feeds from the standard output. The result is then compared to "1" using the equality operator. If both values are equal, the path exists. Otherwise, the path does not exist.

7. Under the When clause, add an execution step as shown below. This execution step calls the \texttt{/system/compute-string} function to build the string value that should be returned when the path exists. Notice that the value embeds the FlowForce expression \{path\}. This expression references the input parameter defined in a previous step.

8. Under the Otherwise clause, add an execution step as shown below. This execution step calls the \texttt{/system/compute-string} function to build the string value that should be returned when the path does not exist. Notice that the value embeds the FlowForce expression \{path\}. This expression references the input parameter defined in a previous step.

9. Under Execution Result, declare the return type as \texttt{string}. 
10. Under Service, click to select the **Make this job available via HTTP** check box, and enter **CheckPathService** as name of the service. For more information, see [Exposing Jobs as Web Services](#).

![](Image)

11. Under "Credentials", select an existing credential record or specify a local credential. For more information, see [Credentials](#).

12. Click **Save**.

### Running the job

To run the job, do one of the following:

- Go to Home, and then click **Show all active triggers and services**. Next, click the job's URL displayed in the "Info" column.
- Enter http://127.0.0.1:4646/service/CheckPathService in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the Configuration page, change the address accordingly.
- If you set the optional **Host name** field of FlowForce Server from the Setup Page, you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the **Make this job available via HTTP** check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the Service - Use permission to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the Service - Use permission to the user **Anonymous**, see [How Permissions Work](#).

Since this job has arguments, you will be prompted to supply them when you access the Web service in the browser.
If the job executes successfully, the browser displays the output of the job, for example:

Path C:\ exists.

If the job fails, the browser displays a "Service execution failed" message. In this case, check the log of the job in FlowForce Server to identify the error, see Viewing the Job Log.
13.3 Copy Files

This example shows you how to copy multiple files on the local file system with the help of a FlowForce Server job.

Let's assume that you would like to copy all the files from directory C:\FlowForceExamples\CopyFiles\Source to a new directory C:\FlowForceExamples\CopyFiles\Target. (On a UNIX system, please adjust the paths accordingly.) To achieve the goal, we will use a "for-each" step that iterates through all the files in a directory, and then invoke the /system/filesystem/copy function for each item in the loop.

Prerequisites

- Required licenses: FlowForce Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container used in this example is accessible to any authenticated user).
- This job copies files from directory C:\FlowForceExamples\CopyFiles\Source to directory C:\FlowForceExamples\CopyFiles\Target. Make sure to create these directories on the local file system before creating the job. Also, make sure that the source directory contains a few files to test the job.

Creating the job

Log on to FlowForce Server and navigate to a container where you have permission to create new jobs. For consistency with other examples, this tutorial uses the /public/Examples container—if you don't have this container yet, create it using the Create | Create Container command.

In the /public/Examples container, create a new job. Enter a job name (for example, "CopyFiles"), and, optionally, a job description.

In order to iterate over items in a list, FlowForce Server provides a "for-each" execution step. Such a step iterates over a sequence (list) of items up to and including the last item in the sequence. In this example, our sequence of items will be the list of files in the source directory. To create the required list, click New Execution Step and type /system/compute next to "Execute function". You can also select this path from the drop-down list, as illustrated below.
Next, enter the following expression in the Expression field:

```
list-files("C:\FlowForceExamples\CopyFiles\Source\*.*")
```

Next, enter a name for the list in the Assign this step’s result to field (in this case, the name is list). This makes it possible to easily refer to the newly created list of files in a subsequent step. Your first execution step should now look as follows:

The expression above invokes the list-files expression function. The function takes a path as argument (in this case, C:\Source\*.*) and returns the list of files (or directories) at the given path. Notice that the path contains the wildcard *.* to select all the files in the directory. If necessary, you can adjust the wildcard to select only specific file extensions, for example *.txt. For more information about expressions in FlowForce, see FlowForce Expressions.

You can now proceed to creating the actual "for-each" iteration step. Click New For-Each step and type list in the "in sequence" box. (This refers to the list created in the previous execution step.)
Tip: You could also copy the expression to the "in sequence” box of the "for-each” step and thus get rid of the first execution step altogether.

Next, click the button and add a new execution step inside the "for-each” step. This step will invoke the /system/filesystem/copy function for each item in the loop, as illustrated below.

As shown above, the copy function is called with the following arguments:

- The Source is the current item (file) in the loop. You can either type {item} in the Source box or click the Set to button and select item.
- The Target is the target path. In this example, the path is entered as is; however, you could also supply it as an argument to the job.
The Overwrite option is enabled, meaning that if a file with the same name already exists in the source directory, it will be overwritten. To prevent this from happening, click the button.

For the sake of simplicity, we will not set the other two arguments in this example. For further information, see the description of the \system/filesystem/copy function.

The job created so far now includes all the required processing steps, but it has no trigger yet. To trigger the job at recurring time intervals, you could use a timer trigger, see Timer Triggers. Or you can monitor the source directory for changes and trigger the job by means of a file system trigger, see File System Triggers. Finally, you can trigger the job on demand, as a Web service call.

In this example, we will trigger the job on demand, by clicking a URL in the browser (in fact, this invokes the job as a Web service). To turn the job into a Web service, select the Make this job available via HTTP... check box and enter the name of the Web service.

Finally, the job needs your credentials to run. Therefore, enter your operating system username and password (not your FlowForce Server username and password) in the "Credential" section, as shown below. Alternatively, if you created standalone credentials previously, as described in Defining Credentials, you can select them using the Select existing credential option.

Running the job

To test the job, do one of the following:

- Go to Home, and then click Show all active triggers and services. Next, click the job's URL displayed in the "Info" column.
- Enter http://127.0.0.1:4646/service/CopyFilesService in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the Configuration page, change the address accordingly.
- If you set the optional Host name field of FlowForce Server from the Setup Page, you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the Make this job available via HTTP check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to
FlowForce Server.

Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the Service - Use permission to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the Service - Use permission to the user Anonymous, see How Permissions Work.

Upon successful execution, the job will copy all the files from the source to the target directory. Otherwise, a "Service execution failed" error is displayed in the browser. If you see this error, check the log of the job for further information, see Viewing the Job Log. Possible causes may include incorrect credentials, incorrect file paths, insufficient permissions on the file system, and others. For example, the job fails if the Overwrite check box is not selected and the target directory already contains a file with the same name, as illustrated below:

![Job execution log](image)
13.4 Create a Job from a MapForce Mapping

This example shows you how to create a FlowForce Server job from a MapForce mapping. First, you will deploy a demo mapping file from MapForce to FlowForce Server. Once the mapping is deployed to FlowForce Server, you will create a server job from it. The job will be configured to run daily at a specific time.

Prerequisites

- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port.
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container used in this example is accessible to any authenticated user).
- The mapping job created in this example generates an XML file every time when it runs. Therefore, on the operating system where FlowForce Server runs, you must have rights to create files in some directory (this example uses the C:\FlowForceExamples\Mapping directory).

Demo files used

The mapping file used in this example is called CompletePO.mfd, and it is available at the following path on the computer where MapForce is installed: \Documents\Altova\MapForce2022\MapForceExamples. Note that the "MapForceExamples" directory is created when you run MapForce for the first time (but not earlier).
The demo mapping illustrated above takes three XML files as input and produces a single XML file as output. In this example, the input XML files will be included automatically in the package deployed to FlowForce Server. Other mappings may require extra preparation steps before you can deploy them, as described in Deploying Mappings to FlowForce Server.

Creating the job

Deploying a mapping means that MapForce organizes the resources used by the mapping into a single package and sends it through HTTP (or HTTPS, if configured) to FlowForce Server.

To deploy the mapping to FlowForce Server:

1. Open the CompletePO.mfd file in MapForce.
2. If you haven’t done so already, set the transformation language of the mapping to "Built-in".
4. In the Server and Port text boxes, enter the server name and port of the Web administration interface (for example, 127.0.0.1 and 8082, if the FlowForce Web Server service is listening on the same machine at the default port). Change these values if you have configured a different address and port, see Defining the Network Settings.
5. In the User and Password text boxes, enter your FlowForce Server user name and password.
6. Select either Directly from the Login drop-down list, or leave the <Default> option as is.
If Directory Service integration is enabled, enter your domain user name and password, and then select your domain name from the Login drop-down list. For more information, see Changing the Directory Service Settings.

7. For consistency with other examples, we will be deploying the mapping to the /public/Examples container. Click Browse and change the container path to /public/Examples. The /public/Examples container must already exist if you followed the previous examples; otherwise, you can create it by clicking Create Container in the dialog box below:
8. Select the **Open web browser to create new job** check box.
9. Click **OK** to deploy the mapping.

When deployment finishes, the FlowForce Server Administration Interface opens in your web browser, and a partially filled in job page is displayed. The mapping function itself is saved at the container path specified earlier. This concludes the deployment part.

**Creating the job**

After you have deployed the mapping file to FlowForce Server as described above, the browser displays a partially filled job page. The first execution step is created automatically with some pre-filled parameters.
Create job in /public

Job name: CompletePO.job
Job description:

Job Input Parameters

Execution Steps

You can also create the job by opening the function's page (/public/Examples/CompletePO.mapping), and then clicking Create job.

To configure the job:

1. Change the default job name from "CompletePO.job" to something more descriptive, for example, "GeneratePurchaseOrder". This is an optional step, but it may be necessary if the name is already used by some other job in the same container.
2. Fill in the first execution step created by default as follows:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>This field points to the mapping function deployed earlier; leave it as is.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>The Customers, Articles, and ShortPO fields contain the respective XML files pre-packaged into the job.</td>
</tr>
<tr>
<td></td>
<td>The CompletePO field specifies the path of the output file. By default, it is CompletePO.xml. The path is relative to the working directory, as further described below.</td>
</tr>
</tbody>
</table>
Create a Job from a MapForce Mapping

3. Under “Triggers”, click **new Timer**.
4. Next to "Run", set the timer to run **Daily** every 1 days. Next to "Start", select a date and time when the job must start, for example:

   ![Triggers interface](image)

   - **Run**: daily
   - **Repeat**: every 1 day(s)
   - **Start**: 2020-09-17 12:55:00
   - **Time zone**: Europe/Vienna

5. Under “Credentials”, select an existing credential record or specify a local credential. For details, see [Credentials](#).

   ![Credential interface](image)

   - **User name**: someuser
   - **Password**: **********

6. Click **Save**.
Running the job
At the time and date specified in the trigger, FlowForce Server executes the mapping job. If the job executes successfully, the file generated as a result (CompletePO.xml) becomes available in the C:\FlowForceExamples\Mapping directory. To see whether the job executed successfully, refer to the job log.
13.5 Use a Job as Step of Another Job

This example shows you how to use a previously defined job as a step of another job. Since this example requires a previously created job, you should complete the Creating a Job from a MapForce Mapping example before completing this example.

As you may recall from the Creating a Job from a MapForce Mapping example, the GeneratePurchaseOrder job generates an XML file in a temporary folder every time it runs. This example shows you how to do the following:

1. Create a job that copies the file generated by the mapping to another directory. We will call this job CopyOutput.
2. Modify the GeneratePurchaseOrder job to include the CopyOutput job as an additional execution step.

Prerequisites

- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user).
- The mapping job created in this example copies files from one directory to another. Therefore, on the operating system where FlowForce Server runs, ensure that both directories exist and that you have rights to create files in both directories. This example uses the C:\FlowForceExamples\Mapping and C:\FlowForceExamples\Archive directories.
- Complete the steps described in the Creating a Job from a MapForce Mapping example.

Creating the job

1. Click Configuration, and then navigate to the /public/Examples container. The public/Examples container should already exist if you followed the previous examples; otherwise, create it using the Create | Create Container command.
2. Click Create, and then select Create Job.
3. Enter the name of the job (in this example, "CopyOutput").

<table>
<thead>
<tr>
<th>Create job in /public/Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Job name:</strong> CopyOutput</td>
</tr>
<tr>
<td><strong>Job description:</strong> Copies the output of a mapping to another directory</td>
</tr>
</tbody>
</table>

4. Under "Execution steps", add the first execution step, with the following settings:
As you may have noticed, the job we just created does not have any trigger. We did not define any trigger because we will call this job from another job.

### Adding the "CopyOutput" job as a step of another job

1. Open the **GeneratePurchaseOrder** from the **/public/Examples** container.
2. Under "Execution Steps", click **new Execution step** to add a new step after the existing one.
3. Next to "Execute function", browse for the **CopyOutput** job created earlier. The execution steps should now look as follows:

   ![Execution Steps Diagram]

4. Update the time trigger, and then click **Save**.
5. At the time entered in the trigger, FlowForce Server executes the job and copies the **CompletePO.xml** file to the specified directory and renames it to **PurchaseOrder.xml**. To see whether the job executed successfully, refer to the **job log**.
13.6 Create a Directory Polling Job

This example shows you how to monitor a directory for changes with the help of a file system trigger created in FlowForce Server (see also File System Triggers). Whenever a new XML file is added to the directory, FlowForce Server executes a mapping job that takes the XML file as input parameter. The output of the mapping job is then moved to an archive directory.

Prerequisites

- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port.
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user).
- The mapping job created in this example copies files from one directory to another. Therefore, on the operating system where FlowForce Server runs, ensure that both directories exist and that you have rights to create files in both directories. This example uses the C:\FlowForceExamples\DirPolling and C:\FlowForceExamples\Archive directories.

Demo files used

- ShortApplicationInfo.mfd — the MapForce mapping from which the FlowForce Server job will be created
- ApplicationsPage.xml — the XML instance file to be supplied as input to the mapping.

Both files are available at the following path on the machine where MapForce is installed: <Documents>\Altova\MapForce2022\MapForceExamples\.

What the mapping does

The MapForce mapping used in this example (ShortApplicationInfo.mfd) is illustrated below. From a FlowForce Server perspective, the important thing is that the mapping takes an XML file as input, and produces another XML file as output.

Essentially, this mapping converts an XML file (ApplicationsPage.xml) to a different schema and saves it as ShortInfo.xml. The mapping is relatively easy to understand by looking at the topmost connection: for each Item found in the source, it creates an Info item in the target. The other connections are used to copy values from the respective child items. Of particular interest is the dotted connection; in MapForce, this connection is called “Source-driven (Mixed Content)” and it is used because SubSection contains mixed content.
Notice the names of the source and target XML schemas are `SectionedPage` and `ShortInfo`, respectively. As you will see further below, the FlowForce job will have an input and output parameter with the same name, after the mapping is deployed to FlowForce Server.

**Deploying the mapping to FlowForce Server**

The mapping `ShortApplicationInfo.mfd` does not need any special preparation before it is deployed to FlowForce Server. Since both the source and target components are XML files, they will be included automatically in the package deployed to FlowForce Server.

To deploy the mapping to FlowForce, open it in MapForce and run the menu command **File | Deploy to FlowForce Server**.
If FlowForce Server runs on a different host and port, change the connection details above accordingly, see Defining the Network Settings. Also note that the path of the mapping is /public/Examples/ShortApplicationInfo.mapping; this is consistent with previous examples.

Creating the directory polling job

After the mapping is deployed to FlowForce Server, the browser opens and loads the job creation page. As illustrated below, the first execution step is created automatically and it calls the mapping function deployed previously. Notice that the input parameter has the same name as the source MapForce component (SectionedPage), while the output parameter has the same name as the target component (ShortInfo).
Configure the job as follows:

1. In the **Working-directory** box, enter the path to the working directory. This example uses C:\FlowForceExamples\DirPolling as working directory.

   A working directory is a parameter required by execution steps if the job needs a location to unpack any input files or save output files. FlowForce Server also uses the working directory to resolve any relative paths that occur during step execution. When asked to provide a working directory, you should supply a valid path on the operating system where FlowForce Server runs. If you do not supply a working directory when creating the step, FlowForce Server uses a temporary directory.

2. Under “Triggers”, click **new Filesystem trigger**. Notice that FlowForce Server automatically adds a new **triggerfile** parameter under "Input Parameters". You will need to refer to this parameter in a subsequent step.

3. Set the following trigger values:
   - Check: Modified Date
   - File or directory: C:\FlowForceExamples\DirPolling\*.xml
   - Polling interval: 60 seconds

4. Under Execution Steps, supply the **triggerfile** parameter as input value to the **SectionedPage** parameter. To do this, click the **Set to** button next to the SectionedPage parameter, and then...
select **triggerfile**. As a result, the value of the **SectionedPage** parameter changes to \{**triggerfile**\}. The curly braces denote a FlowForce expression and should not be removed.

With the configuration done so far, the trigger will fire whenever **ApplicationsPage.xml** is copied into the working directory. However, since the trigger uses a wildcard (*.xml), it would be fired also when any other XML file changes inside the directory, including the mapping output itself (**ShortInfo.xml**). This is not the intended behavior and could cause errors; therefore, let's add a second step that will move the generated output file to a new directory. Alternatively, you could rename the trigger to C:\FlowForceExamples\DirPolling\ApplicationsPage.xml (in this case, a second step is no longer necessary).

To add the step which moves the output to a new directory, do the following:

1. Add a new execution step, immediately after the previous one.
2. Configure the step as follows (note that the source and destination fields are case-sensitive):

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the <code>/system/filesystem/move</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td><strong>ShortInfo.xml</strong></td>
</tr>
<tr>
<td></td>
<td>We used a relative path because the <strong>Working Directory</strong> parameter is set, see below.</td>
</tr>
<tr>
<td>Destination</td>
<td>This must be an existing file or directory path on the operating system where FlowForce Server runs. Set it to:</td>
</tr>
<tr>
<td></td>
<td>C:\FlowForceExamples\Archive</td>
</tr>
<tr>
<td>Overwrite target</td>
<td>Select this check box. This instructs FlowForce Server to overwrite any file with the same name found at the destination path.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>Leave this parameter as is.</td>
</tr>
<tr>
<td></td>
<td>This Boolean parameter determines what should be the return value of the function if the job fails. If <strong>Abort on error</strong> is FALSE, the function will return Boolean FALSE as well. If <strong>Abort on error</strong> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
<tr>
<td>Working directory</td>
<td>FlowForce will look for all relative file paths in this directory. Set it to:</td>
</tr>
<tr>
<td></td>
<td>C:\FlowForceExamples\DirPolling</td>
</tr>
</tbody>
</table>
Finally, add your operating system credentials with which the job will be executed:

1. Under "Credentials", select an existing credential record or specify a local credential. For more information, see Credentials.
2. Click Save.

Running the job
You can now test the job by copying the file ApplicationsPage.xml to the working directory. When you do this, FlowForce Server executes the mapping job and copies the resulting output file to the archive directory.

To see whether the job executed successfully, refer to the job log.
13.7 Add Error Handling to a Job

This example illustrates how to add error handling to a job. The job used in this example lists the contents of a directory and will be invoked from the browser, as a Web service. You will learn how to configure FlowForce Server to handle the job outcome as follows:

- If the job execution is successful, display the job's output in the browser
- If the job fails to execute due to any reason, send an email notification to a named recipient.
- Whenever the job execution finishes, regardless of the execution status, log the job internal ID to a file on the local system.

In FlowForce Server terms, in this example you create a protected block with two error handling conditions: "On Error" and "Always" (each will handle one of the scenarios mentioned above).

Prerequisites

- Required licenses: FlowForce Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port
- The FlowForce Server mail settings must be configured, see Setting the Mail Parameters
- You need a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user).
- The job created in this example writes output to the disk. Therefore, on the operating system where FlowForce Server is installed, you need read and write access to some directory. This example uses C:\FlowForceExamples\ErrorHandling.

Tips

- Although this example uses Windows paths and commands, you can still test it on other operating systems, if you adapt the paths and the commands accordingly.

Creating the job

1. On the computer where FlowForce Server runs, create a directory that will be the job's working directory. This example uses C:\FlowForceExamples\ErrorHandling.
2. Log in to the FlowForce Server Web administration interface and go to the /public/Examples container. The public/Examples container should already exist if you followed the previous examples; otherwise, create it using the Create | Create Container command.
3. Click Create Job and enter a name for the job you are creating, for example "ListDirectory". The job's description is optional.
4. Under "Job Input Parameters", click the + button, and add a parameter of type "string". At job runtime, the parameter will provide the path of the directory to list. In this example, the name of the parameter is "inputDir"; it will be used in subsequent steps.
6. Under "Execute with error/success handling", click the + button, and choose to add a new execution step, with the following settings:

   Execute function | Browse for the /system/shell/commandline function.
Add Error Handling to a Job

Advancing beyond the basics

425

Job Configuration Examples

Command

Enter the following shell command:

dir {inputDir}

Where \texttt{inputDir} is the name of the parameter created previously. The name is enclosed within curly braces because, at job runtime, its content will be replaced dynamically with the parameter value. For more information, see \href{Embedding Expressions in String Fields}{Embedding Expressions in String Fields}.

Abort on error

Leave this option as is. For more information, see the description of the \texttt{system/shell/commandline} function.

Working directory

Enter the path to the working directory created previously, for example \texttt{C:\FlowForceExamples\ErrorHandling}.

7. Under the "On error" condition, click the \boxed{+} button and choose to add a new execution step, with the following settings:

Execute function

Browse for the \texttt{system/mail/send} function.

From

Enter the email address of the sender, for example \texttt{flowforce@localhost}. Leave this field empty if you have configured the mail settings from the administration page.

To

Enter your email address.

Subject

Enter the subject of the notification email as follows:

\texttt{Job \{instance-id\} has failed}

The part between curly braces is a FlowForce expression which calls the \texttt{instance-id} function to get the unique ID of the current (failed) job instance.

Message body

Type the following:

\texttt{Exit Code: \{string(exitcode(failed-step()))\}}
\texttt{Standard Error: \{content(stderr(failed-step()))\}}
\texttt{Error message: \{error-message(failed-step())\}}

The parts between curly braces are two FlowForce expressions. These expressions get the erroneous output and convert it to a string that will be the body of the email:

- The \texttt{failed-step} function returns the \texttt{result} of the failing step. This is an abstract FlowForce type that, in order to become more useful, must be supplied as argument to the \texttt{exitcode}, \texttt{stderr}, or \texttt{error-message} functions, see below.
- The \texttt{exitcode} function gets the actual exit code of the error from the \texttt{result}, as a number, assuming that there is an exit code.
### The stderr function

The `stderr` function gets the standard error output of the error from the `result`, as a stream, assuming that there is standard error output.

### The error-message function

The `error-message` function gets the text of the FlowForce error message as it appears in the log. It may also return an empty string if there is no error or if it is not technically possible to retrieve the error text.

### The string function

The `string` function converts the numeric exit code to a string (this must be done because the body of the email is of string type).

### The content function

The `content` function converts the error output from a stream to a string.

The `exitcode` and `stderr` functions return a value only if execution produces an exit code and error output, respectively. This is typically the case for errors such as the ones produced by the command line. The `error-message` function is just for informational purpose and is not guaranteed to return the text of the error for every possible job configuration and error condition encountered.

---

8. Click **new error/success handler**, and then select **Always**.

9. Under the "Always" condition, click the **+** button and choose to add a new execution step, with the following settings:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the <code>/system/shell/commandline</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Enter the following shell command:</td>
</tr>
<tr>
<td></td>
<td><code>echo {instance-id()} &gt;&gt; JobLog.txt</code></td>
</tr>
<tr>
<td></td>
<td>On Windows, this command writes the job ID to a file called <code>JobLog.txt</code>. If the file contains data, the new text will be added after the existing data.</td>
</tr>
<tr>
<td>Working directory</td>
<td>Enter the path of the directory created previously (<code>C:\FlowForceExamples\ErrorHandling</code>).</td>
</tr>
<tr>
<td></td>
<td>This directory will be used to resolve the path to the <code>JobLog.txt</code> file.</td>
</tr>
</tbody>
</table>

At this stage, the job should look as follows (provided you did not use different paths or shell commands).
10. To turn the job into a Web service, select the **Make this job available via HTTP** check box and enter the name of the Web service, for example:

![Service](image)

Take notice of the service name; you will need it to call the Web service.

11. Under “Credentials”, select an existing credential record or specify a local [credential](#).  
12. Click **Save**.

**Running the job**

At this stage, you have completed the job configuration. Because this job is exposed as a Web service, you can run it in any of the following ways:
• Go to Home, and then click Show all active triggers and services. Next, click the job's URL displayed in the "Info" column.
• Enter http://127.0.0.1:4646/service/ListDirectoryService in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the Configuration page, change the address accordingly.
• If you set the optional Host name field of FlowForce Server from the Setup Page, you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the Make this job available via HTTP check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

Supposing your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the Service - Use permission to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the Service - Use permission to the user Anonymous, see How Permissions Work.

Since this job takes parameters, you will be prompted to supply a parameter value when you access the Web service from the browser.

If you enter a valid directory path like C:\, for example, the job is executed, and the outcome is displayed in the browser.

Also, each time when you run the job, the ID of the job instance is appended to the contents of the JobLog.txt file, according to the "Always" condition configured previously.

To test the "On Error" condition, change the "inputDir" parameter to some deliberately incorrect value (for example, a path that does not exist). If this case, the browser will display an error and FlowForce Server will send an email to the address specified in the recipient field of the "On Error" handler. For example, the e-mail could look as follows:
As stated previously, the error functions used in this example are not guaranteed to return a value for each and every possible job configuration. Therefore, the level of detail provided by the e-mail depends on your job configuration and the kind of error encountered, and it should not be expected that the **Exit Code**, **Standard Error**, and **Error message** e-mail fields always contain text. The most authoritative reference for the cause of the error is the FlowForce Server log.
### 13.8 Expose a Job as a Web Service

This example illustrates how to create a FlowForce Server job exposed as a Web service. The job executes a mapping designed with Altova MapForce. The mapping queries data from a SQLite database and retrieves only records that match a value supplied as parameter when the Web service is called. You will learn how to deploy the existing mapping from MapForce to FlowForce Server and turn it into a Web service. After completing this example, you will be able to invoke the Web service from a browser.

#### Prerequisites
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port.
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user)
- The mapping job created in this example writes an XML file to a local directory. Therefore, on the operating system where FlowForce Server runs, a writeable directory must exist where the job output will be created. This example uses C:\FlowForceExamples\GetPersonRecords.
- The job used in this example reads data from a SQLite database and does not require installing any database drivers. However, if you would like to use a different database, then the database drivers must be installed not only on the computer where the mapping is designed but also on the server where the job runs. For example, in case of Microsoft Access databases, the Microsoft Access Runtime (https://www.microsoft.com/en-us/download/details.aspx?id=50040) must be installed on the machine where FlowForce Server runs.

#### Demo files used
This example makes use of the following files, available at the following path on the computer where MapForce is installed: ..\Documents\Altova\MapForce2022\MapForceExamples\Tutorial.

- FilterDatabaseRecords.mfd (the MapForce mapping design file)
- Nanonull.sqlite (the SQLite database from which the mapping reads data).

#### What the mapping does
The mapping discussed in this example is called FilterDatabaseRecords.mfd and is available in the "Tutorial" folder of MapForce (..\Documents\Altova\MapForce2022\MapForceExamples\Tutorial).
As illustrated above, the source component is a database which stores user records. The target component is an XML file. The connection from users to row creates one row for each database record extracted from the source. The mapping also contains an input parameter to be supplied at runtime. Double-click the title bar of the input parameter to view its properties:

The mapping also contains a SQL-WHERE component placed between the source and the target. The goal of the SQL-WHERE component is to pass on to the target component only those database records that match the condition last_name LIKE :sqlparam. Again, this is configured from the component properties:
On the mapping, the value of `sqlparam` is obtained by concatenating the input parameter with the `%` character. Therefore, if the caller supplies the input parameter "m" at runtime, then the mapping will retrieve user records whose last name begins with "m".

For more information about designing mappings such as the one discussed in this example, refer to MapForce documentation (https://www.altova.com/documentation).

Preparing the mapping for deployment to FlowForce Server

In the instructions below, the term "source machine" refers to the computer where the MapForce is installed and the term "target machine" refers to the computer where FlowForce Server is installed (this may or may not be the same computer).

Before attempting to deploy the mapping to the target machine, do the following:

1. Make sure that the "FlowForce Web Server" service is configured to listen for client HTTP(S) requests, see Defining the Network Settings. For example, if FlowForce Server is installed on the same computer and is configured with the default settings, then you should be able to access it by typing `http://localhost:8082` in your browser. If FlowForce Server is running on a different computer, make sure that the incoming connections to the specified address and port are not blocked by the firewall.
2. Make sure that the "FlowForce Server" service is also configured to listen for client HTTP(S) requests. This service handles requests to jobs exposed as Web services, see also How It Works. Therefore, in order for the Web service to be accessible to HTTP clients outside of the local host, the "FlowForce Server" service must be configured to listen either on all interfaces, or on a specific address other than the local host. You can check whether this service is configured correctly by accessing the following URL from the browser: http(s)://<host or IP address>:<port>/service/ . When prompted to enter a password, supply the password of your FlowForce Server user account. All jobs that are exposed as Web services (if any) should appear as links directly in the browser window.

3. Verify that the mapping is configured to use relative instead of absolute paths, as follows:

a) Open the FilterDatabaseRecords.mfd mapping in MapForce, right-click the mapping area, and select Mapping Settings from the context menu.

b) If applicable, clear the Make paths absolute in generated code check box.

c) Save the mapping.

File-based databases such as Microsoft Access or SQLite are not deployed to a target machine together with the mapping. Therefore, the SQLite database must be manually copied from the source machine to the target machine. Copy the Nanonull.sqlite database file from the directory ..\Documents\Altova\MapForce2022\MapForceExamples\Tutorial on the source machine to some empty directory on the target machine. In this example, the target directory is C:\FlowForceExamples\GetPersonRecords. Take notice of this path because it will be referenced later from the FlowForce job.

The mapping is now ready for deployment to FlowForce Server. For more information about deploying mappings which include database connections, see Preparing Files for Server Execution.

Deploying the mapping

To deploy the mapping to FlowForce Server:

1. On the File menu, click Deploy to FlowForce Server. If you are deploying the mapping to FlowForce Server on a different machine, change the server address and port from "localhost:8082" to those configured from the FlowForce Server network settings.
2. For consistency with all other examples, we will choose to deploy the mapping to the `/public/Examples` container. Click Browse and change the container path to `/public/Examples`. The `/public/Examples` container must already exist if you followed the previous examples; otherwise, you can create it by clicking Create Container.

3. Select the Open new browser to create new job check box.

4. Click OK.

For reference to all deployment settings, see Deploying Mappings to FlowForce Server.

Creating the FlowForce job

So far, you have deployed the mapping to FlowForce Server and have the job configuration page open in the browser (provided that you selected the check box Open new browser to create new job on the dialog box above). Otherwise, login to the FlowForce Server Web administration interface, open the previously deployed mapping function (it should be in the `/public/Examples` container), and then click Create Job.

To configure the job:

1. Under "Job Input Parameters", create a new input parameter of type string. This value will be provided by callers of the Web service when they invoke the job. Let's name it "LookupValue".
2. Configure the execution step as follows:

- Set the value of the input parameter to the "LookupValue" input parameter created in previous step.
- Set the working directory to C:\FlowForceExamples\GetPersonRecords. Note that this directory must already exist on the file system, and it must already contain the source NaNonull.sqlite database if you followed the previous steps.

3. To turn the job into a Web service, select the Make this job available via HTTP... check box and enter the name of the Web service, for example:

Take notice of the service name; you will need it to call the Web service.

4. Under "Credentials", select an existing credential record or specify a local credential.

**Note:** These are the credentials of your user account on the operating system and not the ones used to access the FlowForce Server Web administration interface. The user account must be able to access the NaNonull.sqlite database file from the working directory; otherwise, the job will fail to execute successfully.

5. Click Save.
Invoking the Web service
At this stage, you have completed the job configuration. Because this job is exposed as a Web service, you can run it in any of the following ways:

- Go to Home, and then click Show all active triggers and services. Next, click the job's URL displayed in the "Info" column.
- Enter http://127.0.0.1:4646/service/GetPersonRecordsService in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the Configuration page, change the address accordingly.
- If you set the optional Host name field of FlowForce Server from the Setup Page, you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the Make this job available via HTTP check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the Service - Use permission to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the Service - Use permission to the user Anonymous, see How Permissions Work.

Since this job takes parameters, you will be prompted to supply a parameter value when you access the Web service from the browser.

If you enter a valid directory path like M, for example, the job will query the database and return only the rows where the person's last name begins with "M", for example:
On job failure, a "Service execution failed" error is displayed in the browser. If you see this error, check the Job Log for further information.
13.9 Post JSON to FlowForce Web Service

This example shows you how to create a FlowForce Web service that accepts POST requests carrying JSON data in the HTTP request body. Secondly, it illustrates how to call the Web service from a client like MapForce.

In this example, the Web service will be configured to accept JSON data. You could also post XML or other content to a service created with FlowForce Server in a similar way as shown below. The Web service is intended to be very simple so it will merely accept JSON data and save it locally without any further processing. It is possible to further extend the job to validate the JSON data with RaptorXML Server, or process it, although this will not be done in this example.

This example specifically illustrates the case when data is posted in the body of the HTTP request, not as a parameter. For an example that invokes a Web service with parameters, see Expose a Job as a Web Service.

Prerequisites


Remarks

FlowForce Server provides a quick way to create the Web service. MapForce Enterprise Edition acts as a client that calls the Web service created with FlowForce Server. You may also use a different client and achieve the same result.

- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port.
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user).
- This job saves input data received by the Web service to a local working directory, C:\FlowForceExamples\PostJson. This directory (or a similar one) must exist on the machine where FlowForce Server runs, and your operating system user account must have rights to write to this directory.

Creating the FlowForce job

Log in to the FlowForce Server Web administration interface, open the /public/Examples container, and then click Create Job. Next, enter a name and, optionally, a description for the Web service you are creating.

Note: The public/Examples container should already exist if you followed the previous examples; otherwise, create it using the Create | Create Container command.
In order for the job to treat the POST data as arbitrary content, it must have exactly one parameter of type `stream`. To create the parameter, click Add parameter, enter a parameter name (in this example, “data”), and select `stream` as data type.

Next, add a new execution step and configure it as follows:

The execution step above calls the FlowForce built-in `copy` function. The expression shown in the “Source” text box converts the input received by the Web service to a file by using the `as-file` expression function (recall that the input parameter was named `data` in a previous step). To obtain this expression automatically, click the button next to the “Source” text box and then select `data`.

The “Target” text box contains an expression that produces a unique file name each time when the job is invoked. To obtain the unique file name, the FlowForce `instance-id` expression function is called; therefore, the JSON file name will look something like “file35.json”, and the number will be different with each job call (corresponding to the ID of that FlowForce job instance). You could also enter a full path, but it is not necessary if the “Working directory” path is set, as it was done in this example. When you set the working directory path, any relative file name will be resolved relative to the working directory path.
The directory C:\FlowForceExamples\PostJSON (or a similar one if you changed the path) must exist and your operating system user account must have rights to write to it.

Under "Service", select the Make this job available via HTTP check box, and enter "PostJsonService" or a similar name for the new Web service. Take notice of the service name; you will need it to call the Web service.

Under "Credentials", select an existing credential record or specify a local credential (see also Credentials). These must be the credentials of the user account on the operating system where FlowForce Server runs.

Note: Do not confuse these credentials with the ones used to access the FlowForce Server Web administration interface.

Click Save. You are now ready to call the new Web service from a client.

Calling the Web service from a browser
You can call the Web service from a browser in any of the following ways:

- Go to Home, and then click Show all active triggers and services. Next, click the job's URL displayed in the "Info" column.
- Enter http://127.0.0.1:4646/service/PostJsonService in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the Configuration page, change the address accordingly.
- If you set the optional Host name field of FlowForce Server from the Setup Page, you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the Make this job available via HTTP check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the Service - Use permission
to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the Service - Use permission to the user **Anonymous**, see *How Permissions Work*.

Because the job was configured to expect a stream as parameter, you are now prompted to enter the parameter value in the browser. Click **Browse** and select the JSON file to be submitted in the POST request.

When you click **Submit**, FlowForce Server processes the job and outputs the response to the browser.

If the job executes successfully, the browser displays "true" and the JSON file is saved to the working directory `C:\FlowForceExamples\PostJson`. Otherwise, if you see an execution error, refer to the job log for more details, see *Viewing the Job Log*.

**Calling the Web service from MapForce**

You can also call the Web service from a client other than the Web browser, for example, from MapForce Enterprise Edition.

1. On the **File** menu, click **New** to create a new mapping.
2. On the **Output** menu, click **Built-in Execution Engine**.
3. On the **Insert** menu, click **Web Service Function**. The Web Service Call Settings dialog box opens.
4. Click **Manual**, choose **POST** as request method, and enter the URL of the web service in the URL box. This is the same URL that was used to test the Web service from the browser.
5. Click the **Edit** button next to "HTTP Security Settings", and select the **Dynamic authentication** check box. This makes it possible to supply the credentials interactively as input parameters to the mapping when the mapping runs. For information about the **Use credential** option, see [Credentials in Mapping Functions](#). Entering the username and password directly in this dialog box is supported only for backward compatibility and is not recommended.

6. Click **OK** to close the dialog box. The mapping now looks as follows:
7. Add to the mapping three input parameters, by selecting the **Insert | Insert Input** menu command. The first two will supply the username and password, respectively. The third will supply the JSON data.

8. Double-click each of the input components above, and enter a design-time execution value to be used for previewing the mapping. For the first two parameters, enter the username and password required to access the Web service—these are necessary to run the mapping, and, for security reasons, it's not recommended to save them in the mapping file. For the parameter that will supply JSON data, enter some sample JSON data to be used for executing this mapping at design time, like the one shown below:

![Create Input dialog](image)

**Note:** The sample JSON data shown here is very short, for demo purposes. When MapForce Server runs the mapping, you can supply the JSON data as input parameter to the mapping from an actual JSON file.
9. Add the output of the mapping, by selecting the **Insert** | **Insert Output** menu command.

10. Drag the `charset-encode` and `mime-entity` functions from the Libraries window and make all the connections as shown below. You will also need to add two constants, by selecting the **Insert** | **Constant** menu command.

In the mapping above, the JSON input is provided to the mapping by means of a simple input component. The `charset-encode` and `mime-entity` functions are MapForce built-in functions that prepare the body of the HTTP request. The status code returned by the Web service is mapped to the result returned by the mapping.

Preparing the body of the HTTP request in an unstructured manner as shown above is just one of the ways to send data in the POST request. For JSON and XML structures, you can enter the JSON or XML schema of the request in the "Web Service Call Settings" dialog box instead. In this case, the body of the Web service component provides mapping inputs (connectors) based on the JSON/XML structure of the request.

You can now execute the mapping with MapForce, by clicking the **Output** tab. If an error occurs, it is displayed in the Messages window. To debug, you may need to check the FlowForce Server log as well (assuming that the POST request reached the server). Otherwise, if execution is successful, the following happens:

1. The HTTP status code "200" is displayed in the **Output** pane.
2. On the server side, the submitted JSON data is written to a file and saved to the `C:\FlowForceExamples\PostJson` directory.

The exact behavior of the mapping in case of an error can be further configured from MapForce. Also, the mapping can be run with MapForce Server, or be deployed to FlowForce Server, and turned into a job or even...
another Web service. For further information, refer to MapForce documentation
13.10 Cache Job Results

This example shows you how to cache the result of a job (referred to as cache producer) and use it in another job (referred to as cache consumer). Both jobs will be exposed as Web services with the following behavior:

- When the cache producer Web service is invoked, it lists recursively the contents of the directory, creates or updates the cache, and then outputs the result in the browser;
- When the cache consumer Web service is invoked, it reads the cache created by the cache producer service and outputs the result in the browser.

Our goal is to compare the execution time of both jobs, and see that the second job executes significantly faster than the first job, since it consumes cached data.

Prerequisites

- Required licenses: FlowForce Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user).

Note: Although this example uses Windows paths and commands, you can still test it on other operating systems, if you change the paths and the commands accordingly.

Configuring the job

1. Click Configuration, and then navigate to the /public/Examples container. The public/Examples container should already exist if you followed the previous examples; otherwise, create it using the Create | Create Container command.
2. Click Create, and then select Create Job.
3. In the Job Name box, enter CachedResult.
4. Under "Execution Steps", add a new execution step with the following settings:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the /system/shell/commandline function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Enter the following shell command:</td>
</tr>
<tr>
<td></td>
<td>dir /s</td>
</tr>
<tr>
<td></td>
<td>On Windows, this command lists recursively the contents of the working directory (see the next setting).</td>
</tr>
<tr>
<td>Working directory</td>
<td>Set the value to a directory on the machine where FlowForce Server runs, for example:</td>
</tr>
<tr>
<td></td>
<td>c:\</td>
</tr>
<tr>
<td></td>
<td>If you would like to use a different directory, choose one that is big enough so that it takes at least 20-30 seconds to list the directory contents recursively.</td>
</tr>
</tbody>
</table>
Assign this step's result to | We will need to refer to the value returned by the execution step in a subsequent step, so it must have a name. For the scope of this example, enter `dir` as value of this field.

5. Under Execution Steps, add a new execution step with the following settings:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the <code>/system/compute</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Enter the following FlowForce Server expression: <code>stdout(dir)</code></td>
</tr>
<tr>
<td></td>
<td>The <code>stdout</code> function converts the raw result returned by the previous execution step into a stream of data (see Step Result Functions).</td>
</tr>
</tbody>
</table>

6. Under "Execution Result", set the return type to `stream`. As you might have noticed, we set it to the same data type as returned by the last execution step of the job. The job should now looks as follows:

7. Under "Caching Result", select the **Cache the result** check box.
8. Select the **Auto-create a new cache consumer job** check box, and then enter `DirectoryListingCachedService` as the name of the Web service.
9. Under "Service", click to select the **Make this job available via HTTP** check box, and enter `DirectoryListingService` as name of the service.

10. Under "Credentials", select an existing credential record or specify a local credential, see [Credentials](#).

11. Click **Save**.

**Running the job**

At this stage, you have completed the configuration of both the cache producer and the cache consumer jobs. To test the performance of the non-cached service (`DirectoryListingService`) in the browser, run the job using any of the following approaches:

- Go to **Home**, and then click **Show all active triggers and services**. Next, click the job's URL displayed in the "Info" column.
- Enter `http://127.0.0.1:4646/service/DirectoryListingService` in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the **Configuration page**[5], change the address accordingly.
- If you set the optional **Host name** field of FlowForce Server from the **Setup Page**[6], you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the **Make this job available via HTTP** check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

> **Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the **Service - Use** permission to this user on the container where the job is, and then access the Web service with the corresponding**
user account. To disable HTTP authentication and make the Web service public, grant the **Service - Use** permission to the user **Anonymous**, see [How Permissions Work](124).

Note that, because the job was configured to list the contents of the C:\ directory recursively, it might take up to several minutes to complete. Refer to the job log to see how long it took for the job to complete, see [Viewing the Job Log](124).

To test the performance of the cache consumer service (**DirectoryListingCachedService**), enter [http://127.0.0.1:4646/service/DirectoryListingCachedService](http://127.0.0.1:4646/service/DirectoryListingCachedService) (or the equivalent URL if your host name and port are configured differently) in the browser's address bar. Since this service consumes the cache rather than executing the directory listing, it is expected to take significantly less time to complete.
13.11 Create a Job from a StyleVision Transformation

This example shows you how to create a FlowForce Server job from a StyleVision transformation. The job will consist of three steps, namely:

1. The first step will execute the StyleVision transformation.
2. Because the transformation returns an array of multiple streams, the second step will access one of the several files created by the transformation, using a FlowForce Server expression.
3. The third step will copy the file to an archive folder.

Prerequisites

- Required licenses: StyleVision Enterprise or Professional edition, StyleVision Server, FlowForce Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container used in this example is accessible to any authenticated user).
- The job created in this example copies files from one directory to another. Therefore, on the operating system where FlowForce Server runs, ensure that both directories exist and that you have rights to create files in both directories. This example uses the following directories:
  - C:\FlowForceExamples\GenerateHtml — this is the job's working directory where all processing happens and relative paths are resolved.
  - C:\FlowForceExamples\Archive — the destination directory to which the HTML file produced by the job will be copied.

Demo files used

The StyleVision Power Stylesheet (.sps) file used in this example processes an XML file and produces output in multiple formats, including HTML. It is called AutoCalc.sps, and is available in the StyleVision "Examples" project, under Examples > Basics > AutoCalc.sps. To open the StyleVision examples project in StyleVision, click Examples on the Project menu.

Deploying the StyleVision transformation to FlowForce Server

First, let's deploy the demo .sps file from StyleVision to FlowForce Server. Deploying an .sps file means that StyleVision organizes the resources used by the transformation into an object and passes it through HTTP (or HTTPS if configured) to FlowForce Server. Once the transformation is deployed to FlowForce Server, you will create a server job from it.

To deploy the StyleVision transformation:

1. Open the AutoCalc.sps file in StyleVision.
2. On the File menu, click Deploy to FlowForce.... If this option is disabled, make sure the Design tab is currently selected. When prompted to save the transformation as PXF file, leave the default settings as is, and click OK.
3. In the **Server** and **Port** text boxes, enter the server name and port of the Web administration interface (for example, `127.0.0.1` and `8082`, if the *FlowForce Web Server* service is listening on the same machine at the default port). Change these values if you have configured a different address and port, see [Defining the Network Settings](#).

4. In the **User** and **Password** text boxes, enter your FlowForce Server user name and password.

5. Select either **Directly** from the **Login** drop-down list, or leave the **<Default>** option as is.

If Directory Service integration is enabled, enter your domain user name and password, and then select your domain name from the **Login** drop-down list. For more information, see [Changing the Directory Service Settings](#).

6. The **Path** text box displays the default path where the transformation will be deployed. For consistency with other examples, click **Browse** and change the path to `/public/Examples/AutoCalc.transformation`. The `/public/Examples` container must already exist if you followed the previous examples; otherwise, you can create it by clicking **Create Container** in the dialog box below.
7. Click OK, and select the **Open web browser to create new job** check box on the "Deploy Transformation" dialog box.
Create a Job from a StyleVision Transformation

8. Click OK to deploy the transformation.

When deployment finishes, the FlowForce Server Administration Interface opens in your web browser, and a partially prefilled job page is displayed. The transformation function itself is saved at the container path specified earlier. This concludes the deployment part.

Creating the job

After you have deployed the .sps file to FlowForce Server as described above, the browser displays a partially filled job page. The first execution step is created automatically with some prefilled parameters.
You can also create the job by opening the function's page (/public/Examples/AutoCalc.transformation), and then clicking **Create job**.

**To configure the job:**

1. Change the default job name from "AutoCalc.transformation.job" to something more descriptive, for example, "GenerateHtml". This is an optional step, but it may be necessary if the name is already used by some other job in the same container.

2. Fill in the first execution step created by default as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute function</td>
<td>This field points to the StyleVision transformation function deployed earlier; leave it as is.</td>
</tr>
<tr>
<td>Parameters</td>
<td>The <strong>InputXml</strong> field contains an XML file that is pre-packaged into the job (<strong>Data.xml</strong>). For the scope of this example, you can leave this option as is. For information about changing input instances, see <strong>Running Mappings and Transformations as Jobs</strong>.</td>
</tr>
<tr>
<td></td>
<td>To declare the <strong>AutoCalc.html</strong> as output file, click the <strong>+</strong> button next to <strong>OutHtml</strong>.</td>
</tr>
</tbody>
</table>
Create a Job from a StyleVision Transformation

The option `GenerateHtmlOutputAsMime` is not set in this example. This option is typically necessary to prepare the HTML body of an e-mail message sent with FlowForce Server. For more information, see the `send-mime` function.

The options `OutRtf`, `OutFo`, `OutPdf`, `OutDocx` declare transformation outputs in the respective formats (RTF, FO, PDF, Docx). These outputs are not used in the current example, so they can be left unchanged.

In the `Working-directory` box, enter the path to a directory where FlowForce Server must save the job output. This example uses `C:\FlowForceExamples\GenerateHtml` as working directory.

Assign this step's result to `Enter output`. This explicitly gives the name "output" to the result returned by the first execution step. We will need to refer to this result subsequently.

3. Click **new Execution step** and configure it as follows:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the <code>/system/compute</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Enter the following FlowForce Server expression:</td>
</tr>
<tr>
<td></td>
<td><code>as-file(nth(results(output), 0))</code></td>
</tr>
<tr>
<td></td>
<td>This expression instructs FlowForce Server to do the following:</td>
</tr>
<tr>
<td></td>
<td>1. Call the expression function results to get the array returned by <code>output</code> declared previously.</td>
</tr>
<tr>
<td></td>
<td>2. Pass this array to function <code>nth</code> to get the first item in the array. Since the array index is zero-based, we are using 0 as second argument of function <code>nth</code>.</td>
</tr>
<tr>
<td></td>
<td>3. Pass the value to the <code>as-file</code> function to declare it as a file.</td>
</tr>
<tr>
<td>Assign this step's result to</td>
<td>Enter <code>html_file</code>. This instructs FlowForce Server that the result returned by the step has the name <code>html_file</code>. We will need to refer to this result subsequently.</td>
</tr>
</tbody>
</table>

4. Click **new Execution step** and configure the step as follows:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the <code>/system/filesystem/copy</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Click <code>Set to</code>, and then select <code>html_file</code>.</td>
</tr>
<tr>
<td>Target</td>
<td><code>C:\FlowForceExamples\Archive\AutoCalc.html</code></td>
</tr>
<tr>
<td>Overwrite</td>
<td>Select the <code>Overwrite</code> check box.</td>
</tr>
<tr>
<td>Working directory</td>
<td><code>C:\FlowForceExamples\GenerateHtml</code></td>
</tr>
</tbody>
</table>
At this stage, the "Execution Steps" section of the job page should look as follows:

5. Under “Triggers”, click **new Timer**.
6. Next to "Run", set the timer to run **Daily every 1 days**. Next to "Start", select a date and time when the job must start, for example:
9. Under "Credentials", select an existing credential record or specify a local credential. For details, see Credentials.
10. Click Save.

Running the job
At the time and date specified in the trigger, FlowForce Server executes the StyleVision transformation job. If the job executes successfully, the AutoCalc.html file becomes available in the C:\FlowForceExamples\Archive directory. To see whether the job executed successfully, refer to the job log.
13.12 Validate a Document with RaptorXML

This example shows you how to create a job which validates an XML Schema file. This example shows probably the easiest way to validate a file, because it does not use conditional error handling and does not write the validation result to a custom log file or to the browser. The validation result will be available only in the FlowForce Server log. For a more complex validation job example, see Validate XML with Error Logging.

The validation job used in this example calls the valany function of RaptorXML Server. The valany function validates a document based on its type. It takes the file to validate as the only mandatory parameter, and it can be used to validate XML files, XML schemas, DTD schemas, and other file types. For a list of RaptorXML functions, see the RaptorXML documentation (https://www.altova.com/documentation).

Prerequisites

- Required licenses: FlowForce Server, RaptorXML (or RaptorXML+XBRL) Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container used in this example is accessible to any authenticated user).

Demo files used

This example job validates the address.xsd file available in the RaptorXML Server installation folder, at the following path: C:\Program Files\Altova\RaptorXMLServer2022\examples\address.xsd.

On a 64-bit Windows running 32-bit RaptorXML Server, the path is C:\Program Files (x86)\Altova\RaptorXMLServer2022\examples\address.xsd, unless you installed RaptorXML Server in a different folder.

You can use any other XML schema file as well.

Creating the job

1. Log in to the FlowForce Server Web administration interface and open the /public/Examples container. The public/Examples container should already exist if you followed the previous examples; otherwise, create it using the Create | Create Container command.
2. Click Create Job. Next, enter a name and, optionally, a description for the job you are creating. This example uses "ValidateSchema" as job name.
3. Click new Execution step.
4. Next to "Execute function", browse for the /RaptorXML/valany function. Note that the mandatory parameter File is shown as an expanded field.
**Execution Steps**

![Execution Steps](image)

**Note:** The `valany` function exists directly under the "RaptorXML" container and also in any container that corresponds to a specific RaptorXML release, for example, "2022". For information about differences between the two, see [Generic versus release-specific RaptorXML functions](#).

5. In the **File** text box, enter the path to the schema file that you want to validate, for example, `C:\Program Files\Altova\RaptorXMLServer2022\examples\address.xsd`.

6. Under "Triggers", click **new Timer** and create a trigger that will run the job at a specific time in future. For details, see [Timer Triggers](#).

7. Under "Credentials", select an existing credential record or specify a local credential. For details, see [Credentials](#).

8. Click **Save**.

**Running the job**

The job will run at the date and time specified in the trigger. To see whether the job executed successfully, refer to the **job log**. Specifically, in the **Instance Log** page, an entry like the one below indicates successful validation:

```
file:///C:/Program%20Files/Altova/RaptorXMLServer2021/examples/address.xsd: runtime="0ms" result="OK" cmd="valxsd"
```

If the file did not validate, the job execution is considered failed (since at least one of the steps has failed), so an error is reported in the log. In this case, the logged entry displays `result="Fail"` along with details about the validation error.
13.13 Validate XML with Error Logging

This example shows you how to create a job which validates an XML file against a schema. If the job fails due to any reason, the error details will be written to a log file. For validation, we will use the `valxml-withxsd` function of RaptorXML Server running under FlowForce Server management. Note that, for the error logging part, the technique illustrated in this example is not dependent on RaptorXML Server and can be applied to other job types.

**Note:** The RaptorXML Server functions become available in FlowForce Server after RaptorXML Server is installed.

In this example, the job will be defined as a Web service, so that you can trigger it on demand, by accessing a URL from the browser. You can also add to the job a timer (or file system) trigger, similar to how this is done in other examples. You could even add to the same job a combination of a trigger and a Web service. This way, the job will run not only as defined by the trigger rules, but also on demand, when the Web service is called.

**Prerequisites**
- Required licenses: FlowForce Server, RaptorXML (or RaptorXML+XBRL) Server
- The *FlowForce Web Server* and *FlowForce Server* services must be listening at the configured network address and port
- Your FlowForce Server user account has permissions to one of the containers (by default, the `/public` container used in this example is accessible to any authenticated user).
- The job created in this example generates a log file every time when it runs. Therefore, on the operating system where FlowForce Server runs, you must have rights to create files in some directory (this example uses the `C:\FlowForceExamples\ValidateXml` directory).

**Demo files used**

The XML file validated in this example is available in the RaptorXML Server installation folder, at the following path: `C:\Program Files\Altova\RaptorXMLServer2022\examples\NanonullOrg.xml`.

On a 64-bit Windows running 32-bit RaptorXML Server, the path is `C:\Program Files (x86)\Altova\RaptorXMLServer2022\examples\NanonullOrg.xml`, unless you installed RaptorXML Server in a different folder.

You could also use any other XML file for validation.

**Creating the job**

1. Log in to the FlowForce Server Web administration interface and open the `/public/Examples` container. The `/public/Examples` container should already exist if you followed the previous examples; otherwise, create it using the *Create* | *Create Container* command.
2. Click *Create Job*. Next, enter a name and, optionally, a description for the job you are creating. This example uses "ValidateXml" as job name.
3. Under "Job Input Parameters", click the button and create a new parameter of type "string as file", for example:

4. Under "Execution Steps", click the button, and then select new error/success handling step.

5. Under "Execute with error/success handling", click the button, and choose to add a new execution step, with the following settings:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the /RaptorXML/valxml-withxsd function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Next to the XML File parameter, click Set to and select the inputFile job input parameter declared earlier.</td>
</tr>
</tbody>
</table>

6. Under the "On error" condition, click the button and choose to add a new execution step, with the following settings:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the /system/compute function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Set the value of Expression to:</td>
</tr>
</tbody>
</table>
The parts between curly braces are two FlowForce expressions. This expression gets the output, converts it to a stream and then writes it to a file on the disk:

- The `failed-step` function returns the `result` of the failing step. This is an abstract FlowForce type that, in order to become more useful, must be supplied as argument to the `exitcode`, `stderr`, or `error-message` functions, see below.
- The `stdout` function gets the standard output from the `result`, as a stream, assuming that there is standard output.
- The `as-file` function creates a file from the stream. The path will be specified in a subsequent step.

### Assign this step's result to
Enter a value which will uniquely identify the result of this job, for example, `MyResult`. By doing this, you are declaring this value as a variable, so that you can use it in a subsequent step.

7. Click the button to add a new execution step after the previous one, with the following settings:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>Browse for the <code>/system/filesystem/copy</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Next to the <code>Source</code> parameter, click <code>Set to</code> and select the <code>MyResult</code> variable declared earlier.</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td>Next to the <code>Target</code> parameter, type the path where the log will be saved (in this example, the path is <code>C:\FlowForceExamples\ValidateXml\error.log</code>).</td>
</tr>
<tr>
<td></td>
<td>Select the check box next to the <code>Overwrite</code> parameter. The log file is generated each time the job runs, so this ensures that the job does not fail when the log file already exists.</td>
</tr>
<tr>
<td></td>
<td>Set the <code>Working Directory</code> parameter to <code>C:\FlowForceExamples\ValidateXml</code>.</td>
</tr>
</tbody>
</table>

The "On Error" branch of the job should now look as follows:
8. Under "Service", select the **Make this job available via HTTP** check box, and enter **ValidateXmlService** as name of the service.

9. Under "Credentials", select an existing credential record or specify a local credential. For details, see **Credentials**.

10. Click **Save**.

### Running the job

To run the job, do one of the following:

- Go to **Home**, and then click **Show all active triggers and services**. Next, click the job's URL displayed in the "Info" column.
- Enter **http://127.0.0.1:4646/service/ValidateXmlService** in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the **Configuration page**, change the address accordingly.
- If you set the optional **Host name** field of FlowForce Server from the **Setup Page**, you can execute the web service call directly from the job configuration page, by clicking the **button adjacent to the **Make this job available via HTTP** check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

---

Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the **Service - Use** permission to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the **Service - Use** permission to the user **Anonymous**, see **How Permissions Work**.

Since this job was configured to expect a parameter as input, the browser displays a form where you can enter the path to the XML file that is to be validated.
Enter an XML file path in the text box (for example, C:\Program Files\Altova\RaptorXMLServer2022\examples\NanonullOrg.xml), and click Submit.

If the job executes successfully (that is, if it returns the exit code 0), the browser displays the standard output of the job, for example:

```
file:///C:/Program%20Files/Altova/RaptorXMLServer/examples/NanonullOrg.xml: runtime="16ms" result="OK"
```

If the job returns an exit code other than 0 (for example, due to an incorrect path, validation errors, and so on), the browser displays a "Service execution failed" message and the output is written to the C:\FlowForceExamples\ValidateXml\error.log file. In the event that the log file was not generated, check the job log to identify the error. It may be the case, for example, that the /system/filesystem/copy function has failed because you have no permission to write to the target path.
### 13.14 Run XSLT with RaptorXML

This example shows you how to run an XSLT transformation with RaptorXML Server (or RaptorXML+XBRL Server) running under FlowForce Server management. The job will call the `xslt` function of RaptorXML Server. When you configure the job from the FlowForce Server configuration page, there are two ways to supply the parameters to the `xslt` function:

1. By typing key-value pairs (parameter name and value) in text boxes.
2. By entering a FlowForce Server expression.

Both ways are presented in more detail below.

#### Prerequisites
- Required licenses: FlowForce Server, RaptorXML (or RaptorXML+XBRL) Server
- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port.
- Your FlowForce Server user account has permissions to one of the containers (by default, the `/public` container used in this example is accessible to any authenticated user).
- The job created in this example runs an XSLT stylesheet that processes an input XML file. Both files must exist in some directory on the operation system where FlowForce Server runs, and you must have rights to read and write files in this directory. This example uses the `C:\FlowForceExamples\RunXslt` directory.

#### Demo files

The job illustrated below will run an XSLT stylesheet called `transformation.xslt` which takes as input a file called `books.xml`, and two required parameters, "year" and "genre". The exact content of the files is shown in the code listings below. To use these files in the job, save both code listings with the indicated file names to the `C:\FlowForceExamples\RunXslt` directory.

The XSLT stylesheet looks as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
exclude-result-prefixes="xs fn">
  <xsl:output method="xml" encoding="UTF-8" byte-order-mark="no" indent="yes"/>
  <xsl:param name="year" as="xs:string" required="yes"/>
  <xsl:param name="genre" as="xs:string" required="yes"/>
  <xsl:template match="/">
    <library>
      <xsl:attribute name="xsi:noNamespaceSchemaLocation" namespace="http://www.w3.org/2001/XMLSchema-instance" select="'library.xsd'"/>
      <last_updated>
        <xsl:sequence select="xs:string(fn:current-dateTime())"/>
      </last_updated>
      <xsl:for-each select="=./books/book[(fn:string(year) &gt; $year)]">
        <publication>
          <xsl:for-each select="@id">
```

© 2015-2021 Altova GmbH

Altova FlowForce Server 2022
transformation.xslt

The input XML file looks as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="books.xsd">
  <book id="1">
    <author>Mark Twain</author>
    <title>The Adventures of Tom Sawyer</title>
    <category>Fiction</category>
    <year>1876</year>
  </book>
  <book id="2">
    <authorFranz Kafka</author>
    <title>The Metamorphosis</title>
    <category>Fiction</category>
    <year>1912</year>
  </book>
  <book id="3">
    <author>Herman Melville</author>
    <title>Moby Dick</title>
    <category>Fiction</category>
    <year>1851</year>
  </book>
  <book id="4">
    <author>Miguel de Cervantes</author>
    <title>Don Quixote</title>
    <category>Fiction</category>
  </book>
</books>
```
Creating the job

You can create a FlowForce Server jobs to run such an XSLT transformation as follows:

1. Log in to the FlowForce Server Web administration interface and open the /public/Examples container. The public/Examples container should already exist if you followed the previous examples; otherwise, create it using the Create | Create Container command.
2. Click Create Job. Next, enter a name and, optionally, a description for the job you are creating. This example uses "RunXslt" as job name.
3. Click new Execution step.

```
<year>1605</year>
</book>
</books>
```

books.xml

Note: The xslt function exists directly under the "RaptorXML" container and also in any container that corresponds to a specific RaptorXML release, for example, “2022”. For information about differences between the two, see Generic versus release-specific RaptorXML functions.

5. Set the Working directory parameter to C:\FlowForceExamples\RunXslt.
6. Set the XSLT File parameter to transformation.xslt. This file must exist in the working directory.
7. Set the XSLT Input parameter to books.xml. This file must also exist in the working directory.
8. Set the Parameters parameter as follows:

   a. Click the button next to Parameters. This expands a sub-section within the page, where you can add each parameter name and value individually.

   b. Click the button for each new parameter that you need to add. To run the XSLT in this example, you will need to enter the parameters as follows:
9. Under "Service", select the **Make this job available via HTTP** check box, and enter **RunXsltService** as name of the service.
10. Under "Credentials", select an existing credential record or specify a local credential. For details, see [Credentials](#).
11. Click **Save**.

This concludes the job configuration part.

**Supplying XSLT parameters as expression**

In the job configuration above, you have supplied the parameter to the `xslt` function using text boxes. Note that there is a second way to do this, by entering a FlowForce Server expression in the **Parameters** text box, for example:

```
list(\"year\",\"1900\"),\"genre\",\"Fiction\")
```

To use this second approach, click **Set to** next to **Parameters**, and then click `<Expression>`. Make sure that you type the expression very carefully in order to avoid parsing errors. The expression calls the `list` expression function and builds a list of two key-value pairs. In each key-value pair, the key is the parameter name and the value is the parameter value. Importantly, the parameter values are again enclosed within single quotes.

To switch back to the text box layout, click **Set to** next to **Parameters**, and then click `<Value>`.

**Running the job**

To run the job, do one of the following:

- Go to **Home**, and then click **Show all active triggers and services**. Next, click the job's URL displayed in the "Info" column.
- Enter `http://127.0.0.1:4646/service/RunXsltService` in the browser's address bar. Note that this URL works only if the **FlowForce Server** service listens at the default host address and port name. If you have defined other host and port settings in the [Configuration page](#), change the address accordingly.
If you set the optional **Host name** field of FlowForce Server from the Setup Page, you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the **Make this job available via HTTP** check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

---

**Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the **Service - Use** permission to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the **Service - Use** permission to the user **Anonymous**, see [How Permissions Work](#).**

---

If the job executes successfully, the browser displays the output of the job, for example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="library.xsd">
  <last_updated>2020-10-22T17:39:13-02:00</last_updated>
  <publication>
    <id>2</id>
    <author>Franz Kafka</author>
    <title>The Metamorphosis</title>
    <genre>Fiction</genre>
    <publish_year>1912</publish_year>
  </publication>
</library>
```

If the job fails, the browser displays a "Service execution failed" message. In this case, check the FlowForce Server **job log** to identify the error.
13.15 Generate PDFs from XML Files

This example illustrates how to create a FlowForce Server job which takes as input multiple XML files and returns as output multiple PDF files. The FlowForce Server job will invoke both MapForce Server (to generate the XML output from multiple source XML files) and StyleVision Server (to convert the XML output to PDF).

This example requires a basic understanding of how MapForce mappings and StyleVision transformations work. If you are completely new to StyleVision and MapForce, it is recommended to read first the “Tutorials” chapters of MapForce and StyleVision documentation, respectively:

- Quick Start Tutorial (MapForce)
- Quick Start Tutorial (StyleVision)

Prerequisites

- Required licenses:
  - MapForce Enterprise or Professional edition. This tool enables you to design a mapping transformation (.mfd file) that converts (in this example) XML files from one schema to another.
  - MapForce Server or MapForce Server Advanced Edition. This tool enables you to run the mapping on a server, as a job.
  - StyleVision Enterprise or Professional edition. This tool enables you to design a stylesheet (.sps file) that converts an input XML file to a PDF file.
  - StyleVision Server. This tool enables you to run the transformation on a server, as a job.
  - FlowForce Server. This tool provides the means to run the transformations above as a scheduled or on demand job, change inputs if necessary, and monitor execution.

- The FlowForce Web Server and FlowForce Server services must be listening at the configured network address and port.

- Your FlowForce Server user account has permissions to one of the containers (by default, the /public container used in this example is accessible to any authenticated user).

- The job created in this example generates multiple files on the disk. Therefore, on the operating system where FlowForce Server runs, you must have rights to create files in some directory. This example uses the directory C:\FlowForceExamples\GeneratePdfs.

Demo files used

This example makes use of the following sample files, available at the following path:
<Documents>\Altova\MapForce2022\MapForceExamples.

- MultipleInputToMultipleOutputFiles.mfd (the MapForce mapping file)
- PersonListWithGrouping.sps (the StyleVision transformation file)
- Nanonull-Branch.xml, Nanonull-HQ.xml (the input XML files)

What the MapForce mapping does

As illustrated below, the mapping consists of a source component (Altova_Hierarchical), a target one (PersonList), and various intermediary MapForce built-in functions used to build miscellaneous strings to be written to the output.
The mapping takes as input any XML file that begins with "Nanonull-", from the directory `<Documents>\Altova\MapForce2022\MapForceExamples`. This is defined in the source MapForce component settings (in MapForce, right-click the header of the Altova_Hierarchical component illustrated below, and select Properties from the context menu). Notice that "Input File" is set to Nanonull-*.xml, where the asterisk is a wildcard. Literally, the input is any file which begins with "Nanonull-" and has the .xml extension.

The target component, PersonList, is configured to generate file names dynamically based on the file name of the source XML file. This is defined by right-clicking the File/String button at the top of the component, and then selecting Use Dynamic File Names Supplied by Mapping menu option. The connection to the "File <dynamic>" node means that a new file will be created for every value in the source. The remove-folder
The function is meant to get only the file name (without the folder) from the source path. This is then passed as value to the top **concat** function, which builds a string like `Persons-<Source filename>`.

The second **concat** function builds a string like *Generated by Altova...* followed by the complete path to the mapping file. The result is written as a comment in the target XML file.

The third **concat** function uses the output of the **count** function to build a string that indicates how many person records have been mapped from the source. Again, the result is written as a comment in the target XML file.

Finally, the connection to the target **Person** node copies people data from the source to the target. An individual connection exists for each child element of **Person** that must be mapped.

In addition to this, the target component is configured to convert the generated output to PDF, for each XML generated file. Right-click the header of the target component, select **Properties**, and notice that the **StyleVision Power StyleSheet file** text box specifies a relative path to a StyleVision .sps stylesheet. The latter performs the actual conversion of XML to PDF (further discussed below).

To preview the output of this mapping directly in MapForce, click the **Output** tab available under the mapping pane. To preview the PDF result of the StyleVision transformation, click the **PDF** tab. You will notice that multiple XML's (or PDFs, respectively) are generated in the Output pane, for example:

At this stage, it is recommended to save one of the two output XML files to the disk (since, by default, MapForce generates temporary files). The file will act as a sample (working XML) if you would like to open and test the StyleVision power stylesheet in StyleVision (see next section). To save an output file, first click the **Output** tab, and then, on the **Output** menu, click **Save Output File**.
What the StyleVision transformation does

Run StyleVision and open the **PersonListWithGrouping.sps** transformation file. Recall that this file is in the same directory as the MapForce mapping discussed above, and it is referenced by the target MapForce component.

The StyleVision .sps stylesheet illustrated above uses a single XML as source and creates a PDF document from it. The PDF document consists of a heading ("h2"), an introductory paragraph, a table populated dynamically, and an ending paragraph. The heading and the introductory paragraph contain static text, while the table and the ending paragraph are populated from the nodes of the source XML file, as indicated by the wrapping tags.

To preview this transformation directly in StyleVision, follow the steps below:

1. In the **Design Overview** pane, next to **Working XML**, click 🅱️.
2. Select Assign Working XML File and browse for the XML output file saved previously from MapForce (see previous section).
3. Click the PDF tab.

Importantly, the .sps stylesheet is agnostic with the respect to the actual name or origin of the source XML file; it simply processes the XML file provided as input (as long as it conforms to the specified XML schema), and creates a PDF out of it. In order to automate this stylesheet so that it generates multiple PDF files, it will need to be deployed to FlowForce Server, as shown further below.

Deploy the files to FlowForce Server

So far, you have become familiar with the purpose of both the MapForce mapping and the StyleVision transformation used in this example. For more information about designing MapForce mappings and StyleVision stylesheets, refer to the documentation of these products (https://www.altova.com/documentation.html).

To make automation possible, both files must now be deployed to FlowForce Server. As specified in the "Prerequisites" section above, FlowForce Server must be licensed and running, and both MapForce Server and StyleVision Server must be licensed and running under FlowForce Server management. On Windows, you can use the verifylicense command of each server product to check the status of its license. On other operating systems, the job execution will fail with an error message if the license is not found or valid.

To deploy the StyleVision stylesheet to FlowForce Server:

1. On the File menu, click Deploy to FlowForce. (If this command is grayed out, switch to the Design tab first.)
2. When prompted that the design file will be saved as PXF (Portable XML Form) format, click OK.
3. When prompted to select the desired files to be included in the deployed package, leave the default settings as is. Although only PDF is generated in this example, including other outputs will save you time later if you change your mind and want to generate additional formats like HTML and RTF.

4. When prompted, fill in the connection details to FlowForce Web Server. For simplicity, in the image below, the transformation is deployed to the local machine on port 8082, through plain HTTP. It is also possible to specify a remote address and deploy the files through an SSL-encrypted connection, provided that FlowForce Web Server has been configured to accept such connections, see Defining the Network Settings. The user and password values are illustrated below for the root FlowForce.
account; however, any other FlowForce user account may also be used, provided that it has permissions to write data to the specified path. In this example, the **Open browser to create new job** check box has been deliberately left unchecked, because creating and configuring the job will be a separate step discussed further below.

5. For consistency with other examples, it is recommended to use the target path `/public/Examples/PersonListWithGrouping.transformation`.

To deploy the MapForce mapping to FlowForce Server:

- On the **File** menu, click **Deploy to FlowForce Server**. Filling in the connection details illustrated below works the same way as discussed above for StyleVision. Again, for consistency with other examples, it is recommended to use the target path `/public/Examples/MultipleInputToMultipleOutputFiles.mapping`. 
After the files were successfully deployed, the corresponding entries will appear in the specified FlowForce container (in this case, "/public/Examples") when you log on to FlowForce Server:

Note that the entries above are not jobs yet; they are now FlowForce functions from which actual jobs have yet to be created, as shown below.

Create and configure the FlowForce job

Now that the MapForce mapping and the StyleVision transformation have been deployed to FlowForce Server, they can be used to create the required job, as follows:
1. Navigate to the FlowForce /public/Examples container and click the function MultipleInputToMultipleOutputFiles.mapping deployed previously. Notice that the source component of the MapForce mapping discussed at the very beginning of this example has now become an input parameter to the FlowForce function. Also, it has a default value which is the path to the instance XML files processed by the mapping. This value can be overridden later if necessary. The “Working-directory” parameter was added automatically by FlowForce; its role will be clarified in the next steps.

2. Click Create Job.
3. Enter a name and optionally a description for the job you are creating.
4. Configure the “Execution Steps” part of the job as shown below.
5. Under "Service", select the "Make this job available via HTTP..." check box and enter the name of the Web service that will trigger the job on demand, for example "GeneratePdfsService". If you prefer to run the job as a scheduled job, or as a file system trigger, set the appropriate triggers (see Managing Triggers).

6. Under "Credential", enter the username and password of the operating system user account (the job will be executed as this user). Be careful not to confuse this password with the password of the FlowForce Web administration interface (see also Credentials).
7. Click **Save**.

To understand how the job actually works, let’s have a closer look at the “Execution Steps” section of the job. The first execution step calls the mapping deployed previously. It looks for any XML file that begins with “Nanonull-” in the working directory. In this example, the working directory is **C:\FlowForceExamples\GeneratePdfs**.

The output returned by the first execution step represents the data returned by the mapping. It has been explicitly named **output**, in order to make it possible to refer to it in a subsequent step.

The second step of the job is a “for-each” step. Notice how the “for-each” step uses a FlowForce expression **results(output)** to get access to the data returned by the first step (that is, the output returned by the mapping). Specifically, the expression calls the function **results()** which takes as argument the output returned by the previous step, see also **Step Result Functions**. For an introduction to FlowForce expressions, see **FlowForce Expressions**.

The “for-each” step consists of two smaller execution steps:

1. The first step calls the **/system/compute** built-in function to convert the mapping output into an actual file (generically named **file**). Importantly, the output of the **results(output)** expression is a stream, not a file. If the mapping returns multiple outputs (as in this case), the mapping output represents a sequence of streams. For this reason, a FlowForce expression function **as-file** is used to convert the current stream (the one that is being iterated) into an actual file.

2. The second step calls, for each stream that is being iterated, the StyleVision transformation deployed previously. Namely, with each iteration, StyleVision Server is called, an XML file is supplied as input, and a PDF file is returned as output. The FlowForce expression **{filename(file)}.pdf** creates the actual PDF file name on the disk. This operation takes place in the working directory specified by the “Working-directory” parameter.

**Note:** In this example, the same working directory is used by both steps—the one which executes the mapping and the one which runs the StyleVision transformation. In some cases, it may be necessary to specify separate working directories, in order to avoid file name collision or job execution errors.

**Running the job**

To prepare the input data for the job, copy the **Nanonull-Branch.xml** and **Nanonull-HQ.xml** from `<Documents>\Altova\MapForce2022\MapForceExamples` to the working directory (**C:\FlowForceExamples\GeneratePdfs**). This way, the first step of the job gets some input XML files to read data from when the job runs.

To run the job, do one of the following:
Go to Home, and then click **Show all active triggers and services**. Next, click the job's URL displayed in the "Info" column.

Enter [http://127.0.0.1:4646/service/GeneratePdfsService](http://127.0.0.1:4646/service/GeneratePdfsService) in the browser's address bar. Note that this URL works only if the FlowForce Server service listens at the default host address and port name. If you have defined other host and port settings in the Configuration page, change the address accordingly.

If you set the optional **Host name** field of FlowForce Server from the Setup Page, you can execute the web service call directly from the job configuration page, by clicking the button adjacent to the **Make this job available via HTTP** check box. The button is not displayed otherwise.

If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

Supplying your FlowForce Server user credentials for HTTP authentication is only for testing purposes. For production, it is recommended that you create a new FlowForce user, grant the **Service - Use** permission to this user on the container where the job is, and then access the Web service with the corresponding user account. To disable HTTP authentication and make the Web service public, grant the **Service - Use** permission to the user **Anonymous**, see [How Permissions Work](#).

On successful job execution, the PDF files generated by the job will be available in the working directory `C:\FlowForceExamples\GeneratePdfs`. The browser displays "Cannot output the job" even in case of successful execution (this is expected, since the job produces PDF files which cannot be output to the browser). If the job fails to execute for any reason, the browser will display a "Service execution failed" message. In this case, check the **error log** of the job. To troubleshoot issues, you may need to verify again all the prerequisites listed at the top of this page.
14 Glossary

The glossary section includes the list of terms pertaining to FlowForce Server.
14.1 C

Cache consumer
A job that fetches data from the cache if it is available (in other words, it "consumes" the cache). If the cached result is not available, the consumer executes the job and retrieves the actual job result.

Cache producer
A job that populates cache with data (in other words, it "produces" the cache).

Choose step
A step of type "Choose step" lets you define the conditions under which other job steps should be executed. Choose steps have the following structure:

```
When {some expression}
  Execute (some step)
Otherwise
  Execute (some other step)
```

Under each "Choose step", you can nest other Choose steps (sub-conditions), for example:

```
When {expression}
  When {expression}
    Execute (step)
  Otherwise
    Execute (step)
Otherwise
  Execute (step)
```

Any number of conditional steps can be defined.

Within any `When / Otherwise` pair, FlowForce Server executes only the condition that is true; the other condition is ignored.

Container
As the name of the term implies, a container is data packaged together. In FlowForce Server, containers can be roughly compared to folders on an operating system. Containers can contain any of the following: jobs, credentials, functions, and other containers. By setting permissions on containers, you can control who can view or access the data inside them. Organizing data into containers and setting up the relevant permissions for each container is a good security practice.

Credentials
A credential object stores authentication information. This is typically the combination of user name and password associated with a user account on the operating system where the FlowForce Server job runs, but it can also be a set of HTTP or FTP credentials.
14.2  E

Error/Success Handling step
When a step of a job fails, FlowForce Server aborts the job. You can use steps of type “Error/success handling step” to perform clean-up actions before the job finally exits (see Handling Step Errors).

Execution result
In FlowForce Server, a step’s execution result defines what is returned after the step is executed (for example, a file, or some text). When working with jobs, you can explicitly declare a step’s execution result to be of a specific data type (such as String or Boolean), or be discarded. You typically need to declare the data type of the execution result if you intend to use it in other jobs, or if you want to cache the result.

Execution step
A step of type “Execution step” lets you execute a specific FlowForce function. Available functions include the built-in functions delivered with FlowForce Server, deployed MapForce mappings or StyleVision transformations, and the execution steps of other jobs.
14.3  F

For-Each step
A step of type "For-each step" lets you iterate through a sequence (for example, a list of files within a directory) and repeat an execution step any number of times. "For-Each" steps has the following structure:

For each item in sequence (sequence expression)
   Execute (step)

FlowForce executes the step until it finishes looping through all items of the sequence expression.

Function
In the context of a job, a function is an instruction understood by FlowForce Server that performs some operation on the target file system. A function can be one of the following:

- A built-in FlowForce function (see Built-in Functions)
- A StyleVision transformation
- A MapForce mapping
- The execution step of a job

Most functions have input parameters. Any mandatory input parameters must be supplied by the caller in order for the step to execute successfully.
14.4 Input Parameters

In the context of a FlowForce Server job, input parameters represent some arbitrary information supplied to the job based on which the job will execute in a particular way. Job input parameters are similar to function arguments in a programming language. Input parameters can be of various types (for example, file or directory references, text, numbers, Boolean values, and others). For some job types, FlowForce Server creates the input parameters automatically.
14.5  J

Job
A job is a core concept in FlowForce Server. It represents a task or a sequence of tasks to be executed by the server. Jobs can be as simple as one-step tasks such sending an email. However, you can also create jobs that perform multiple actions and pass the result (for example, a file) as parameter to another job. A job consists of input parameters, steps, triggers, and other settings.

Job instance
A job instance is not the same as a job. When you configure a FlowForce job from the job configuration page, you create in fact a job configuration. Every time when the defined trigger criteria for a job apply, an instance of the job starts running.
14.6 P

Password policy
FlowForce Servers uses password policies to help administrators manage the complexity of user passwords. A password policy is a set of minimum requirements that a user password must meet in order to be valid (for example, at least $N$ characters long).

Permission
Permissions control user access to containers. Like privileges, permissions can be granted both to users and to roles. Therefore, if a user is a part of a role, any permissions granted to the role will automatically apply to the user as well.

By default, permissions set on a container are inherited from the parent container. For example, let's assume that container A has a child container B. Users who have permission to access container A will have by default permission to access container B as well. However, an administrator can redefine the permissions of any user or role at every level of the container hierarchy.

Privilege
Privileges define what users can do in FlowForce Server (for example, set own password, read users and roles, stop any job, and so on). Privileges are different from permissions in the sense that permissions control user access to containers, whereas privileges are effective globally across FlowForce Server. The following simple rule might help you distinguish quickly between privileges and permissions: privileges are global, permissions are local.

Like permissions, privileges can be assigned both to individual users and to roles. Therefore, when users log on to FlowForce Server, their set of effective privileges is determined by:

a) the privileges they have been assigned directly
b) the privileges assigned to any roles that the user is member of.
14.7 Queue

Queue
An execution queue is a "processor" of jobs; it controls how job instances run. In order to run, every job instance is assigned to a target execution queue. The queue controls how many job instances (of all the jobs assigned to the queue) can be running at any one time and the delay between runs. By default, the queue settings are local to the job, but you can also define queues as standalone objects shared by multiple jobs. When multiple jobs are assigned to the same execution queue, they will share that queue for executing.

Queues benefit from the same security access mechanism as other FlowForce Server configuration objects. Namely, a user must have the "Define execution queues" privilege in order to create queues, see also How Privileges Work. In addition, users can view queues, or assign jobs to queues, only if they have appropriate container permissions (not the same as privileges), see also How Permissions Work. By default, any authenticated user gets the "Queue - Use" permission, which means they can assign jobs to queues. To restrict access to queues, navigate to the container where the queue is defined, and change the permission of the container to "Queue - No access" for the role authenticated. Next, assign the permission "Queue - Use" to any specific roles or users that you need. For more information, see Restricting Access to the /public Container.
RaptorXML Server

Altova RaptorXML Server (also called RaptorXML for short) is Altova’s third-generation, super-fast XML processor, optimized for the latest standards and parallel computing environments. Designed to be highly cross-platform capable, the engine takes advantage of today’s ubiquitous multi-core computers to deliver lightning-fast processing of XML. RaptorXML is available in two editions: (i) RaptorXML Server and (ii) RaptorXML+XBRL Server. The RaptorXML+XBRL Server edition includes support for validating and processing XBRL (eXtensible Business Reporting Language) documents, in addition to XML.

RaptorXML+XBRL Server

RaptorXML+XBRL Server is a special edition of RaptorXML Server, which includes support for validating and processing XBRL (eXtensible Business Reporting Language) documents, in addition to XML.

Role

Roles are named sets of privileges that help enforce security based on the business need. The typical role-based security involves at least two roles: an administrator and a standard user. Each role is defined by the privileges granted to that role. For example, administrators can change their own password and that of other users, whereas standard users can change only their own password. You can assign roles to users and revoke roles from users as necessary.
Step

Steps define what the FlowForce Server job must actually do (for example, delete a file, execute a MapForce mapping, or send an email). In its simplest form, a step is just an operation with either failed or successful outcome, and it requires a function to be executed (see *Execution step*). However, there are also steps that provide means to execute other steps conditionally or in a loop (see *Choose step*, *Error/success Handling step*, and *For-Each step*). You can create as many steps as required within the same job, and you can set the order in which they must take place.
14.10  Trigger

When you create a job, you must specify conditions (or criteria) that will trigger the job. These conditions are known as triggers. FlowForce Server monitors any defined triggers and executes the job whenever the trigger condition is met.
14.11 U

User

Users are persons who log on to FlowForce Server to configure jobs, deploy MapForce or StyleVision transformations, or manage the FlowForce Server. The actions available to users in FlowForce Server depend on the following:

a) Their assigned permissions or privileges.

b) The permissions and privileges assigned to any roles that users are members of.
14.12 W

Web administration interface
The FlowForce Server Web administration interface allows you to administer the server and configure jobs. You can access the Web administration interface from a Web browser at the configured address and port.

Working directory
A working directory is a parameter required by execution steps if the job needs a location to unpack any input files or save output files. FlowForce Server also uses the working directory to resolve any relative paths that occur during step execution. When asked to provide a working directory, you should supply a valid path on the operating system where FlowForce Server runs. If you do not supply a working directory when creating the step, FlowForce Server uses a temporary directory.
<%APPNAME%%, migrating to a new machine, 52

A

Active Directory,
   integration with FlowForce Server, 90
ADO,
   database connections, 226
ADO.NET,
   database connections, 226
Altova ServiceController, 35
Application pools, 255
Assigning a license to FlowForce Server on Linux, 43
Assigning a license to FlowForce Server on macOS, 48
Assigning a license to FlowForce Server on Windows, 37
Authentication,
   HTTP, 25, 178
   Windows domain, 25, 178

B

Backup, 105

C

Cache consumer,
   definition, 483
Cache producer,
   definition, 483
Choose step,
   definition, 483
Command line interface,
   getting help on, 261
Consumer,
   definition, 483
Container,
   definition, 483
   restricting access to /public, 134
Containers,
   creating, 129
   moving, 129
   overview of, 127
   renaming, 129
   setting permissions on, 132
   viewing permissions of, 131
Credentials,
   adding, 139
   definition, 483

D

Data migration, 105, 106
Data recovery, 105, 106
Data types,
   in FlowForce, 221
Default time zone,
   configuration of, 88
Deinstallation, 28
Digital certificates,
   managing on Windows, 78
   trusting on Linux, 75
   trusting on Mac, 75
   trusting on Windows, 75
Directory polling job,
   example of, 418

E

Environment variables,
   setting, 255
Error handling,
   adding to a job, 424
Error logging,
   adding to a job, 460
Error/success handling step,
   definition, 484
Execution result,
   definition, 484
Execution Step,
  definition, 491
Expressions, 335

**F**

File System Triggers,
  overview of, 174
FlowForce Server,
  application data folder, 102
  architecture, 21
  basic job concepts, 17
  basic security concepts, 19
  built-in functions, 279
  changing the language of, 274
  command line interface, 257
  configuration of, 53
  glossary, 482
  localization of, 104, 266
  log on to, 25
  migrating to latest version of, 271
  new features in, 14
  setting the network address and port of, 60, 63
  setup page, 55
  Web administration interface, 23
FlowForce Server Function,
  definition, 485
FlowForce Web Server,
  setting the network address and port of, 60, 63
For-each step,
  definition, 485
FTP,
  credentials, 139, 141
Function,
  definition, 485

**G**

Global Resources,
  using in FlowForce Server, 242

**H**

Handling errors,
  in jobs, 154, 159
Host name,
  setting for FlowForce Server, 60
  setting for FlowForce Web Server, 60
HTTP,
  increasing the limit of the request body, 63
HTTP Triggers,
  overview of, 176
HTTPS connections,
  configuring FlowForce to accept, 60, 63

**I**

INI files,
  configuration of, 63
Input Parameters,
  adding, 145
  built-in, 145
  definition, 486
  removing, 145
Installation of FlowForce Server, 27
Installation on Linux, 39
Installation on macOS, 45
Installing LicenseServer on Linux, 40
Installing LicenseServer on macOS, 46
Installing LicenseServer on Windows, 33
Installing on Windows, 28
Installing on Windows Server Core, 29
  service properties, 32
  SSL webserver properties, 31
  webserver properties, 30
item,
  as FlowForce data type, 221

**J**

Java,
  configuration, 226
JDBC,
JDBC,
   database connections, 226

Job,
   definition, 487

Jobs,
   caching results of, 182, 446
copying, 144
creating, 143
creating from MapForce mapping, 407
creating from StyleVision transformations, 450
credentials, 139, 141
declaring the return type of, 171
defining queue settings of, 185
duplicating, 144
exporting to another FlowForce Server instance, 190
exporting to file, 190
importing from file, 190
monitoring execution, 201
status reference, 204
stopping, 207
viewing the log of, 209

K

Key/value pairs,
   passing to XSLT sheets, 465

L

LDAP,
   integration with FlowForce Server, 90

License for FlowForce Server,
   assigning on Linux, 43
   assigning on macOS, 48
   assigning on Windows, 37
LicenseServer versions, 33, 40, 46
Licensing FlowForce Server on Linux, 41
Licensing FlowForce Server on macOS, 47
Licensing FlowForce Server on Windows, 35
Licensing of FlowForce Server, 27
Linux,
   installation on, 39
   licensing FlowForce Server on, 41
   starting services on, 99

M

Mac,
   starting services on, 100
   stopping services on, 100
   trusting server certificates on, 75
macOS,
   installation on, 45
   licensing FlowForce Server on, 47
Mail parameters,
   configuration of, 89
MapForce Server,
   calling from a job, 470
   integration with, 225, 236
migratedb command, 106
Migrating FlowForce Server to a new machine, 52

N

Network connections, 34

O

ODBC,
   database connections, 226

Password policies,
   creating, 136
   overview of, 136

Password policy,
   definition, 488
PDF files,
generating, 470
Permission,
definition, 488
Permissions,
list of, 124
overview of, 124
Post-Licensing Tasks on Windows, 51
Postponed steps,
definition of, 147
example, 162
overview of, 162
Privilege,
definition, 488
Privileges,
inheritance, 118
list of, 118
overview of, 118
viewing reports about, 121

R

RaptorXML functions,
creating jobs with, 458, 460
RaptorXML Server,
definition, 490
integration with, 252
RaptorXML+XBRL Server,
definition, 490
Register FlowForce Server with LicenseServer on Linux, 43
Register FlowForce Server with LicenseServer on macOS, 48
Register FlowForce Server with LicenseServer on Windows, 36
Resources,
creating, 242
editing, 242
result,
as FlowForce data type, 221
results,
as FlowForce data type, 221
Role,
definition, 490
Roles,
assigning roles to, 116
assigning to users, 115
built-in roles, 114
creating, 112
renaming, 115
root user,
resetting the password of, 116

S

Service configuration, 34
Set retry count,
as job configuration option, 157
Setup,
on Linux, 39
on macOS, 45
on Windows, 28
Setup of FlowForce Server, 27
SQLite,
as job data source or target, 226
SSL,
encrypting connections with, 60, 63
SSL Encryption,
decrypting the private key, 81
enabling for FlowForce Server, 83
enabling for FlowForce Web Server, 81
introduction, 69
private key requirements, 81
Start FlowForce Server on Linux, 42
Start FlowForce Server on macOS, 47
Start FlowForce Server on Windows, 35
Start LicenseServer on Linux, 42
Start LicenseServer on macOS, 47
Start LicenseServer on Windows, 35
Step,
definition, 491
Steps,
creating, 147
example of using in jobs, 415
handling errors in, 154
StyleVision Server,
calling from a job, 470
integration with, 225, 236
Timer Triggers,
  overview of, 173
Tool files,
  options, 255
Trigger,
  Definition, 492
triggerfile parameter,
  purpose of, 145
Triggers,
  creating, 172
  deleting, 172

UNC,
  syntax in paths, 188, 279
Uninstalling, 28
upgradedb command, 106
Upgrading FlowForce Server on Windows, 50
Users,
  built-in roles, 114
  creating, 111
  renaming, 115

Web administration interface,
  definition, 494
Web Services,
  authentication, 178
  exposing jobs as, 178, 430
Windows,
  installation on, 28
  licensing FlowForce Server on, 35
  referring to network paths on, 279
  starting services on, 101
  stopping services on, 101
  trusting server certificates on, 75
  upgrading FlowForce Server on, 50
Windows domain users,