Table of Contents

1 Introduction 13
  1.1 What’s New ........................................................................................................... 14
  1.2 Basic Concepts ...................................................................................................... 17
  1.3 Security Concepts ................................................................................................. 19
  1.4 How It Works .......................................................................................................... 21
  1.5 Web Administration Interface ............................................................................... 24
  1.6 Logging on to FlowForce Server ........................................................................... 27

2 Installation 29
  2.1 System Requirements ............................................................................................ 29
  2.2 Processor Cores and Licenses ............................................................................... 30
  2.3 Linux ..................................................................................................................... 31
    2.3.1 Installing FlowForce Server ........................................................................... 31
    2.3.2 Migrating to FlowForce Server 2020 ............................................................... 33
    2.3.3 Viewing Currently Installed Products .............................................................. 34
    2.3.4 Uninstalling FlowForce Server ....................................................................... 34
    2.3.5 Installation Directory ..................................................................................... 34
  2.4 macOS ................................................................................................................... 35
    2.4.1 Installing FlowForce Server ........................................................................... 35
    2.4.2 Migrating to FlowForce Server 2020 ............................................................... 36
    2.4.3 Uninstalling FlowForce Server ....................................................................... 37
    2.4.4 Installation Directory ..................................................................................... 37
  2.5 Windows ................................................................................................................ 38
    2.5.1 Installing FlowForce Server ........................................................................... 38
    2.5.2 Windows Service Configuration ....................................................................... 40
    2.5.3 Network Configuration .................................................................................... 41
    2.5.4 Migrating Data Manually ............................................................................... 43
    2.5.5 Installation Folder ........................................................................................... 43
  2.6 Migrating Altova Servers ...................................................................................... 44
3 Configuring the Server

3.1 FlowForce Server Application Data.........................................................49
3.2 Opening the Setup Page........................................................................51
3.3 Defining the Network Settings.................................................................53
  3.3.1 Configuration File Reference..............................................................55
3.4 Setting up SSL Encryption.....................................................................60
  3.4.1 Signing SSL Certificates with a Certificate Authority.........................60
  3.4.2 Creating Self-Signed SSL Certificates...............................................63
  3.4.3 Enabling SSL for FlowForce Web Server..........................................72
  3.4.4 Enabling SSL for FlowForce Server................................................73
  3.4.5 Enabling SSL between FlowForce Web Server and FlowForce Server........75
3.5 Setting the Default Time Zone...............................................................77
3.6 Setting Mail Parameters..........................................................................78
3.7 Changing the Directory Service Settings...............................................79
3.8 Changing the Logging Settings...............................................................82
3.9 Starting and Stopping Services (Linux)................................................83
3.10 Starting and Stopping Services (macOS)...............................................84
3.11 Starting and Stopping Services (Windows)...........................................85
3.12 Localizing FlowForce Server..................................................................86
3.13 Setting Environment Variables..............................................................87

4 Distributed Execution

4.1 Distributed Execution Terminology.......................................................90
4.2 Operating FlowForce Server in "Master" Mode.....................................91
4.3 Converting FlowForce Server to "Worker" Mode.....................................92
4.4 Cluster Installation Options on Windows.............................................95
4.5 Setting up Distributed Execution.........................................................97
4.6 Terminating the "Worker" Mode............................................................101

5 Managing User Access

5.1 Users and Roles.....................................................................................103
  5.1.1 Creating Users...................................................................................103
6.10 Handling Data Types in Steps ................................................................. 154
6.11 Referring to Windows Network Paths .................................................... 157
6.12 Managing Triggers .................................................................................. 158
  6.12.1 Timer Triggers ................................................................................ 159
  6.12.2 File System Triggers ........................................................................ 160
  6.12.3 HTTP Triggers ................................................................................. 162
6.13 Exposing Jobs as Web Services ................................................................. 164
6.14 Caching Job Results ................................................................................. 167
6.15 Defining Queue Settings ......................................................................... 170

7 Importing and Exporting Configuration Data ............................................ 171
  7.1 Worked Examples .................................................................................. 177

8 Monitoring Job Execution .......................................................................... 180
  8.1 Monitoring Execution of Distributed Jobs .............................................. 181
  8.2 Job Statuses ......................................................................................... 182
  8.3 Stopping Jobs ...................................................................................... 185
  8.4 Viewing the Job Log ............................................................................. 187

9 The FlowForce Expression Language .......................................................... 188
  9.1 Computing an Expression .................................................................... 189
  9.2 The Expression Language Rules ............................................................ 190
  9.3 Embedding Expressions in String Fields .............................................. 194
  9.4 Calling Expression Functions ................................................................. 195
  9.5 FlowForce Data Types ......................................................................... 196
  9.6 Operators ............................................................................................ 199

10 Integration with Altova Product Line ......................................................... 200
  10.1 Preparing Files for Server Execution ................................................... 201
  10.2 Running Mappings and Transformations as Jobs ............................... 207
      10.2.1 Credentials in Mapping Functions ............................................. 211
      10.2.2 Dynamic Authentication ............................................................ 212
11 AS2 Integration .................................................. 226
11.1 Overview: Sending AS2 Data ........................................ 227
11.2 Overview: Receiving AS2 Data ........................................ 229
11.3 AS2 Integration with MapForce and MapForce Server .......... 231
11.4 AS2 Concepts ......................................................... 237
11.5 Configuring AS2 Certificates .......................................... 238
11.6 Configuring AS2 Partners ............................................. 242
11.7 Sending AS2 Messages ............................................... 249
11.8 Receiving AS2 Messages .............................................. 254
11.9 Example: Full AS2 Message Exchange (Simple) ............... 259
11.10 Example: Full AS2 Message Exchange (Advanced) .......... 268

12 Command Line Interface ........................................... 278
12.1 help ........................................................................ 280
12.2 assignlicense ......................................................... 281
12.3 compactdb ............................................................ 282
12.4 createdb ............................................................... 283
12.5 debug ..................................................................... 284
12.6 exportresourcestrings ............................................. 285
12.7 foreground ............................................................ 286
12.8 initdb ................................................................. 287
12.9 install ................................................................. 288
12.10 licenseserver ....................................................... 289
12.11 migratedb ........................................................... 290
12.12 repair ................................................................. 291
12.13 resetpassword ...................................................... 292
12.14 setdeflang (sdl) .................................................... 293
12.15 start ................................................................. 294
12.16 uninstall ........................................................... 295
13 Built-in Functions

13.1 /system .......................................................... 299
  13.1.1 abort .......................................................... 299
  13.1.2 compute ...................................................... 300
  13.1.3 compute-string ........................................... 302

13.2 /system/as2 .................................................... 304
  13.2.1 send .......................................................... 304

13.3 /system/filesystem ........................................... 306
  13.3.1 copy .......................................................... 306
  13.3.2 delete ........................................................ 307
  13.3.3 mkdir ........................................................ 308
  13.3.4 move ........................................................ 309
  13.3.5 rmdir ........................................................ 310

13.4 /system/ftp .................................................... 311
  13.4.1 delete ........................................................ 311
  13.4.2 mkdir ........................................................ 313
  13.4.3 move ........................................................ 314
  13.4.4 retrieve ...................................................... 315
  13.4.5 rmdir ........................................................ 317
  13.4.6 store ........................................................ 319

13.5 /system/mail .................................................. 323
  13.5.1 send .......................................................... 323
  13.5.2 send-mime .................................................. 324

13.6 /system/maintenance ....................................... 329
  13.6.1 archive-log ............................................... 329
  13.6.2 cleanup-files .............................................. 329
  13.6.3 truncate-log ................................................ 330

13.7 /system/shell .................................................. 331
  13.7.1 commandline .............................................. 331
## 14 Expression Functions

### 14.1 Step Result Functions
- **exitcode**: 334
- **failed-step**: 335
- **results**: 335
- **stdout**: 337
- **stderr**: 338

### 14.2 Stream Functions
- **as-file**: 339
- **content**: 340
- **empty-stream**: 340
- **stream-from-string**: 340
- **stream-open**: 341

### 14.3 File System Functions
- **list-files**: 343
- **list-directories**: 343
- **read-lines**: 344

### 14.4 File Path Functions
- **extension**: 345
- **filename**: 345
- **filename-with-extension**: 346
- **parent-directory**: 346

### 14.5 List Functions
- **char**: 348
- **code**: 349
- **from-to**: 350
- **join**: 350
- **length**: 352
- **list**: 353
- **nth**: 353
- **slice**: 354

### 14.6 String Functions
- **concat**: 355
14.6.2 contains ........................................................................................................ 355
14.6.3 ends-with ..................................................................................................... 356
14.6.4 find-all ......................................................................................................... 357
14.6.5 number ......................................................................................................... 357
14.6.6 split ............................................................................................................... 358
14.6.7 starts-with .................................................................................................... 358
14.6.8 string ............................................................................................................. 359
14.6.9 string-join .................................................................................................... 359
14.6.10 string-length ............................................................................................. 360
14.6.11 substring ................................................................................................... 360
14.6.12 trim ............................................................................................................. 361
14.6.13 trim-start .................................................................................................. 362
14.6.14 trim-end .................................................................................................... 362

14.7 Boolean Functions .......................................................................................... 363
14.7.1 all ................................................................................................................ 363
14.7.2 any ............................................................................................................... 363
14.7.3 false ............................................................................................................ 364
14.7.4 if .................................................................................................................. 364
14.7.5 not ............................................................................................................... 365
14.7.6 true ............................................................................................................. 365

14.8 Runtime Information Functions ....................................................................... 366
14.8.1 instance-id ................................................................................................. 366
14.8.2 log ............................................................................................................... 366
14.8.3 slot-number ............................................................................................... 368

14.9 AS2 Expression Functions .............................................................................. 369
14.9.1 as2-disposition .......................................................................................... 369
14.9.2 as2-http-status ........................................................................................... 369
14.9.3 as2-mdn-serialize ....................................................................................... 370
14.9.4 as2-message-id .......................................................................................... 370
14.9.5 as2-partner-local-name ............................................................................. 370
14.9.6 as2-partner-remote-name ......................................................................... 371
14.9.7 as2-success ............................................................................................... 371
14.9.8 as2-signed .................................................................................................. 371

14.10 MIME Expression Functions ......................................................................... 373
14.10.1 add-mime-header ...................................................................................... 373
14.10.2  add-mime-headers.............................................................................................................. 374
14.10.3  current-message-id.............................................................................................................. 374
14.10.4  get-mime-content-disposition-param.................................................................................. 375
14.10.5  get-mime-content-id............................................................................................................. 376
14.10.6  get-mime-content-type-param.............................................................................................. 376
14.10.7  get-mime-header.................................................................................................................... 377
14.10.8  get-mime-headers.................................................................................................................. 378
14.10.9  get-stream-filename................................................................................................................ 378
14.10.10 is-file ...................................................................................................................................... 378
14.10.11 is-mime-content-type........................................................................................................... 379
14.10.12 mime-content-encode.......................................................................................................... 380
14.10.13 mime-flatten........................................................................................................................ 380
14.10.14 mime-multipart.................................................................................................................... 381
14.10.15 mime-multipart-related....................................................................................................... 382
14.10.16 mime-multipart-from-list.................................................................................................... 383
14.10.17 mime-parse.......................................................................................................................... 383
14.10.18 mime-split-multipart........................................................................................................... 384
14.10.19 new-message-id..................................................................................................................... 384
14.10.20 reset-mime-headers.............................................................................................................. 385
14.10.21 set-mime-content-disposition.............................................................................................. 385
14.10.22 set-mime-content-id.............................................................................................................. 386
14.10.23 set-mime-header..................................................................................................................... 386
14.10.24 set-mime-headers.................................................................................................................. 387

15  Job Configuration Examples 388

15.1  Create a "Hello, World!" Job....................................................................................................... 389
15.2  Check if a Path Exists.................................................................................................................. 391
15.3  Copy Files..................................................................................................................................... 394
15.4  Create a Job from a MapForce Mapping.................................................................................... 399
15.5  Use a Job as Step of Another Job.............................................................................................. 404
15.6  Create a Directory Polling Job.................................................................................................... 407
15.7  Add Error Handling to a Job....................................................................................................... 412
15.8  Expose a Job as a Web Service.................................................................................................... 417
15.9  Post JSON to FlowForce Web Service....................................................................................... 422
15.10 Cache Job Results .................................................................................................................. 429
15.11 Create a Job from a StyleVision Transformation ................................................................. 432
15.12 Validate a Document with RaptorXML ................................................................................. 437
15.13 Validate XML with Error Logging ......................................................................................... 440
15.14 Use RaptorXML to Pass Key/Value Parameter Pairs ......................................................... 445
15.15 Generate Multiple PDFs from Multiple XMLs ...................................................................... 448

16  Glossary ...................................................................................................................................... 459
16.1  C ................................................................................................................................................. 460
16.2  E ................................................................................................................................................. 462
16.3  F ................................................................................................................................................. 463
16.4  I ................................................................................................................................................. 464
16.5  J ................................................................................................................................................. 465
16.6  M ................................................................................................................................................. 466
16.7  P ................................................................................................................................................. 467
16.8  Q ................................................................................................................................................. 468
16.9  R ................................................................................................................................................. 469
16.10 S ............................................................................................................................................... 470
16.11 T ............................................................................................................................................... 471
16.12 U ............................................................................................................................................... 472
16.13 W ............................................................................................................................................... 473

Index ............................................................................................................................................... 474
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 flavours 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. The Advanced Edition of FlowForce Server can send or accept AS2 messages and adds support for distributed execution of jobs on multiple servers running as a cluster.

Last updated: 11 November 2019
1.1 What's New

FlowForce Server 2020

- FlowForce Server jobs that call Web services can now authorize with the service provider using the OAuth 2.0 protocol. To this end, the "credential" entity in FlowForce has been extended to support OAuth 2.0 fields as well, see OAuth 2.0 Credentials.
- 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.

FlowForce Server 2019 R3

- 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.

FlowForce Server 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.
- FlowForce Server Advanced Edition benefits from distributed execution of jobs. It is now possible to set up multiple FlowForce Server instances as a cluster and redistribute job processing workload across multiple machines. See Distributed Execution and Load Balancing.
- 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.

FlowForce Server 2018 R2

- (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).
- In addition to jobs that send AS2 data, you can also create jobs that run AS2 services and receive AS2 data from partners. FlowForce Server will listen to AS2 requests and respond.
address and port, similar to how it accepts HTTP(S) Web service calls. This effectively turns
FlowForce Server into an AS2 server capable to process and store AS2 data, see AS2 Integration.

FlowForce Server 2018

- FlowForce Server Advanced Edition is now available, in addition to existing FlowForce Server. The
  installation package of FlowForce Server Advanced Edition is bundled with the following additional
  products (licensed separately): MapForce Server Advanced Edition, RaptorXML Server, and
  StyleVision Server. The installation package of FlowForce Server is bundled with the following
- FlowForce Server Advanced Edition provides support for sending AS2 (Applicability Statement)
  messages to AS2 trading partners, see AS2 Integration.
- Internal updates and optimizations

FlowForce Server 2017r3

- Both FlowForce Server and FlowForce Web Server can accept not only plain (unencrypted) HTTP but
  also HTTPS (SSL-encrypted) connections. The HTTPS connectivity settings can be defined either from
  the Setup page (see Defining the Network Settings) or by means of configuration files (see
  Configuration File Reference). On Windows, the connectivity settings can also be defined during
  installation, see Network Configuration (Windows Only).
- During FlowForce Server installation on Windows, the Windows services corresponding to FlowForce
  Server and to FlowForce Web Server can be configured to run as a specific operating system user. The
  start-up manner of the service (automatic, manual) can also be set up, see Windows Service
  Configuration.
- A new system maintenance command is available at the command line: compactdb.

FlowForce Server 2017

- Improved behaviour at system shutdown
- Internal updates and optimizations

FlowForce Server 2016 R3

- Steps, triggers, and parameters of a job can be duplicated using the new Duplicate button.
  Also, after you delete a step, trigger, or parameter, an Undo button is available (see Managing
  Input Parameters, Managing Steps, and Managing Triggers).
- Jobs called by other jobs display parameter descriptions as tooltips, if a description is available. For
  information about adding parameter descriptions, see Adding Input Parameters.

FlowForce Server 2016

- Existing jobs can be saved with a new name, acting as templates for new jobs (see Duplicating
  jobs).
- Enhancements to job management: you can select for deletion or export not only specific jobs within a
  container, but also whole containers, including sub-containers (see Importing and Exporting Jobs).
- It is now possible to perform a root password reset from the command line interface (see Resetting the
  Root Password).
- Performance improvements: timeouts of the Web interface due to intensive job run are now
  significantly reduced.
• The built-in `/system/mail/send` function supports SSL/TLS connections to the SMTP server (see Setting Mail Parameters).

FlowForce Server 2015 R4
• The FlowForce Server installer for Mac is now available as a disk image (.dmg) file (see Installing FlowForce Server).
• Greater flexibility to handle the outcome of failed jobs. When the job execution fails, a new "Abort on error" parameter determines whether the job should be aborted, or the function should return false. This parameter applies to functions from the `/system/filesystem`, `/system/ftp`, `/system/mail`, and `/system/shell` containers.
• Performance improvements: large jobs are loaded significantly faster in the job configuration page.

FlowForce Server 2015 R3
• You can now view from the Web administration interface the date and time when an object (such as a job or function) was created or modified, and the user name who modified it (see Understanding Containers).

FlowForce Server 2015
• Windows Active Directory integration
• Configure the complexity of users passwords
• Reports on system privileges
• Job import and export
• Stop running jobs

FlowForce Server 2014 R2
• Enhancements to FlowForce caching
• Ability to rename users and roles
• Ability to rename containers/objects
• New Step expression functions
• List parameters for RaptorXML (key/value parameter pairs)

FlowForce Server 2014
• New server maintenance functions
• Ability to cache job results and use them in other jobs
• Extended RaptorXML support

FlowForce Server 2013 R2
• Integration with RaptorXML Server
• Job flow control allowing the execution of job steps based on conditions
• Ability to repeat execution steps any number of times
• Definition of step variables allowing the results of one step to be used in following job steps
• An expanded set of built-in steps allowing mail notifications, FTP server interaction, and the ability to compute expressions
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 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 creating a job, you must specify the conditions (or criteria) that will trigger the job, 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, or OAuth security details.

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>Client Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web browser</td>
<td>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>).</td>
</tr>
<tr>
<td>MapForce Enterprise or Professional Edition</td>
<td>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.</td>
</tr>
<tr>
<td>StyleVision Enterprise or Professional Edition</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.
1.5 Web Administration Interface

The FlowForce Server Web administration interface is the front end of FlowForce Server from where you 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 sections describe the menu items available in the Web administration interface.

- **Home**
- **Configuration**
- **Log**
- **Administration**
- **Help**

**Note:** Access to resources and actions available from the Web administration interface is driven by a user access control mechanism. This means that you can access and modify configuration data as long as your assigned permissions allow it. Similarly, you can perform actions (and see the corresponding menu items) if you have been granted the corresponding privilege.

**Home**
Displays the list of currently running and recently finished jobs, see Monitoring Job Execution. The “Active Triggers” section displays any currently enabled triggers, see Managing Triggers.

To refresh the list of jobs or triggers, click **Reload Grid**.

**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 further information about containers, see Understanding Containers.

From the Configuration page, you can also manage containers, jobs, credentials, and functions, and set permissions on containers, provided you have the relevant access rights.

**Log**
Displays the server log. The displayed log entries include both server-related and job-related messages and depend on your access rights.

You can narrow the results by time, job name and the minimum severity (info, warning, error).

You can sort the results in ascending or descending order by any of the following criteria:
### Introduction

<table>
<thead>
<tr>
<th>Date</th>
<th>Specifies the date when the entry was logged.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Specifies the severity of the logged entry. Valid values:</td>
</tr>
<tr>
<td></td>
<td>· Info</td>
</tr>
<tr>
<td></td>
<td>· Warning</td>
</tr>
<tr>
<td></td>
<td>· Error</td>
</tr>
<tr>
<td>Module</td>
<td>Specifies the internal FlowForce Server module from where the logged entry originates.</td>
</tr>
<tr>
<td>User</td>
<td>Specifies the name of the user associated with the logged entry.</td>
</tr>
<tr>
<td>InstanceID</td>
<td>Specifies the system instance ID of the logged entry (if applicable).</td>
</tr>
<tr>
<td>Message</td>
<td>Specifies the message text associated with the logged entry.</td>
</tr>
</tbody>
</table>

To refresh the server log, click **Refresh**.

To export the contents of the log, click **Export**. All records that match the filter criteria will be exported as a 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. The maximum number of records that can be exported is 100,000 (this limitation is imposed by performance reasons). Note that, for the duration of the export, the Web interface remains unresponsive to any other HTTP requests.

### Administration

From the Administration area, you can perform actions to related server configuration and user management. The Administration area consists of the following menu items:

#### Users
- Allows you to create, delete, and maintain users, including changing their privileges, roles, and password policy. For further information, see Managing User Access.

#### Roles
- Allows you to create, delete, and maintain access control roles. For further information, see Users and Roles.

#### Password policies
- Allows you to create, delete, and maintain password complexity policies. For further information, see Password Policies.

#### Reports
- Allows you to view reports on currently assigned user privileges. For further information, see Viewing Privilege Reports.

#### 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 further 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.
# Installation

## 2.1 System Requirements

<table>
<thead>
<tr>
<th>Platform</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>Windows 7 SP1 with Platform Update, Windows 8, Windows 10</td>
</tr>
<tr>
<td>Windows Server</td>
<td>Windows Server 2008 R2 SP1 with Platform Update or newer</td>
</tr>
<tr>
<td>Linux</td>
<td>CentOS 6 or newer</td>
</tr>
<tr>
<td></td>
<td>RedHat 6 or newer</td>
</tr>
<tr>
<td></td>
<td>Debian 8 or newer</td>
</tr>
<tr>
<td></td>
<td>Ubuntu 14.04 or newer</td>
</tr>
<tr>
<td>macOS</td>
<td>macOS 10.12 or newer</td>
</tr>
</tbody>
</table>

The Web administration interface can be accessed with the latest versions of Google Chrome, Mozilla Firefox, and Microsoft Internet Explorer.
2.2 Processor Cores and Licenses

The licensing of Altova server products is based on the number of **physical processor cores** available on the product machine (as opposed to the number of logical cores). 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, you must purchase an 8-core license. You can also combine licenses to achieve the core count. So, two 4-core licenses can be used for an eight-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, will have less processing speed than if all available cores of the computer were utilized.

**Note:** Each Altova server product license can be used for only one client machine at a time—the machine on which the Altova server product is installed—even if the license has unused licensing capacity. 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.

**Single-thread execution**

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 Linux

2.3.1 Installing FlowForce Server

System Requirements

- Linux
  - CentOS 6 or newer
  - RedHat 6 or newer
  - Debian 8 or newer
  - Ubuntu 14.04 or newer

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>LicenseServer</td>
<td>krb5-libs</td>
<td>libgssapi-krb5-2</td>
<td>libgssapi-krb5-2</td>
</tr>
</tbody>
</table>

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 to install FlowForce Server first.

Installing FlowForce Server

1. Download the installation package from https://www.altova.com/download#server to a local directory.
2. Change to the directory where you downloaded the installation package, for example:

   ```
   cd /home/User/Downloads
   ```


   ```
   [Debian, Ubuntu] sudo dpkg --install flowforceserveradv-2020-debian.deb
   [CentOS, RedHat] sudo rpm -ivh flowforceserveradv-2020-1.x86_64.rpm
   ```

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


   ```
   [CentOS 6]: sudo systemctl start flowforcewebserver
   [CentOS 7]: sudo systemctl start flowforcewebserver
   ```
Licensing FlowForce Server

1. Ensure that the latest version of Altova LicenseServer (https://www.altova.com/licenseserver) is installed and running either on the local machine or network.

2. Register FlowForce Server with Altova LicenseServer. For this, you need to know either the host name or the IP address of the computer where LicenseServer runs. For example, if LicenseServer runs on 127.0.0.1, you can register it as follows:

   ```
   sudo flowforceserver licenseserver 127.0.0.1
   ```

   **Note:** The command above assumes you are calling the executable from the program installation directory. On CentOS and macOS, make sure to prepend "/." in front of the executable.

3. Log on to LicenseServer administration interface and assign the license to the machine where FlowForce Server runs.

For more information about licensing, refer to Altova LicenseServer documentation (https://www.altova.com/documentation).

Post-installation tasks

1. Find the URL of the FlowForce Server setup page.

   After you start the FlowForce Web Server for the first time as shown above, it starts on a random port and creates a setup page. If your Linux machine has a Web browser, you can open the setup page in the Web browser, for example:

   ```
   sudo firefox file:///var/opt/Altova/FlowForceServer2020/flowforceweb.html
   ```

   Alternatively, you can extract the setup page URL from the system log:

   **CentOS 7, RedHat 7**

   ```
   sudo grep running /var/log/messages
   ```

   **Debian 8, Ubuntu 16**

   ```
   sudo grep running /var/log/syslog
   ```

   On Ubuntu 14.04 and CentOS 6, you can extract the setup page URL from the FlowForce Web server log file, using the following command:

   **CentOS 6, Ubuntu 14**

   ```
   sudo grep running /var/opt/Altova/FlowForceServer2020/data/ffweb.log
   ```
The output is similar to:

```
FlowForceWeb running on http://127.0.0.1:34597/setup?key=52239315203
```

To access the setup page from another machine on the network, type this link into the address bar of your browser and replace “127.0.0.1” with the host name of your server machine.

```
Make sure that the random port address on which FlowForce Web Server was started is not blocked by your firewall.
```

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, http://localhost:8082, unless you changed the address and port in the previous step) and change the default FlowForce Server root 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), and if you would like to set environment variables for them, see Setting Environment Variables.

### 2.3.2 Migrating to FlowForce Server 2020

#### Prerequisites

- Uninstall the previous version of FlowForce Server (see Uninstalling FlowForce Server). Note that uninstallation does not remove the application data directory which stores your configuration data (such as jobs). The name of the application data directory depends on the major version of FlowForce Server (for example, /var/opt/FlowForceServer2017).
- Install FlowForce Server 2020 (see Installing FlowForce Server). This creates a new application data directory with the default configuration data (for example, /var/opt/FlowForceServer2020).

To migrate data to FlowForce Server 2020:

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

```
[CentOS 6]: sudo initctl stop flowforcewebserver
[CentOS 7]: sudo systemctl stop flowforcewebserver
[Debian 8]: sudo systemctl stop flowforcewebserver
[RedHat]: sudo initctl stop flowforcewebserver
[Ubuntu 14.04]: sudo initctl stop flowforcewebserver
[Ubuntu 16.04]: sudo systemctl stop flowforcewebserver
```

2. Remove or rename the NEW data directory created during installation.

```
sudo rm -rf /var/opt/Altova/FlowForceServer2020/data
```

3. Migrate the EXISTING data by running the `migratedb` command available in the command-line interface of FlowForce Server, for example:

```
[CentOS 6]:     sudo initctl start flowforcewebserver
[CentOS 7]:     sudo systemctl start flowforcewebserver
[Debian 8]:     sudo systemctl start flowforcewebserver
[RedHat]:      sudo initctl start flowforcewebserver
[Ubuntu 14.04]: sudo initctl start flowforcewebserver
[Ubuntu 16.04]: sudo systemctl start flowforcewebserver
```

2.3.3 Viewing Currently Installed Products

To check if an Altova server product is installed:

- Run the following command:

```
[Debian, Ubuntu]   dpkg --list | grep Altova
[CentOS, RedHat]   rpm -qa | grep server
```

2.3.4 Uninstalling FlowForce Server

To uninstall FlowForce Server:

- Run the following command:

```
[Debian, Ubuntu]   sudo dpkg --remove flowforceserveradv
[CentOS, RedHat]   sudo rpm -e flowforceserveradv
```

If you need to uninstall other packages, use the same command as above and supply the package you want to remove as argument. For example:

```
[Debian, Ubuntu]   sudo dpkg --remove licenseserver
[CentOS, RedHat]   sudo rpm -e licenseserver
```

2.3.5 Installation Directory

On Linux, the location of the FlowForce Server installation directory is as follows:

```
/opt/Altova/FlowForceServer2020/bin
```
2.4 macOS

2.4.1 Installing FlowForce Server

System requirements

macOS

macOS 10.12 or newer

Prerequisites

- Ensure that Altova LicenseServer is installed and running either on the local machine or network.
- Perform installation as a user with administrative (root) privileges.
- If you are installing FlowForce Server with other Altova server products, it is recommended to install FlowForce Server first.
- The macOS machine must be configured so that its name resolves to an IP address (that is, you must be able to successfully ping the host name from the Terminal using the command `ping <hostname>`).

Installing FlowForce Server

1. Download the disk image (.dmg) file from the Altova website (https://www.altova.com/download) and click to open it. This causes the installer to appear as a new drive on your computer.
2. Double-click the package (.pkg) file, and go through the successive steps of the installation wizard. These are self-explanatory and include one step in which you have to agree to the license agreement before being able to proceed. When you complete the wizard, the FlowForce Server icon becomes available in Applications. The package is installed in the directory /usr/local/Altova/FlowForceServer2020/.
3. If you have not installed Altova License Server already, or if you want to upgrade to the latest version, double-click the corresponding package (.pkg) file and follow the on-screen instructions. Altova LicenseServer is required to run any Altova server application, including FlowForce Server.

If the name of the Mac machine cannot resolve to an IP address (see 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:

a. Click the link mentioned in the message.
b. In the browser's address bar, replace <hostname> with either localhost or the IP address of your Mac.

5. From the configuration page, register FlowForce Server with Altova LicenseServer.
6. Configure the network address and port of the Web administration interface (see Setting the Network Address and Port).
7. Log on to the Web administration interface (by default, http://localhost:8082, unless you changed the address and port in the previous step) and change the default FlowForce Server root password. The default login name and password is root and root, respectively.
Note: If you have already installed other Altova server products before installing FlowForce Server, copy the .tool file from /etc directory of the corresponding product to the FlowForce Server /tools directory. The .tool file contains the path to the executable of each Altova server product. To copy the tool file, run the `cp` command in the terminal. For example, if you are copying the .tool file from the MapForce Server installation path, run:

```
cp /usr/local/Altova/MapForceServer2020/etc/*.tool /usr/local/Altova/FlowForceServer2020/tools
```

### 2.4.2 Migrating to FlowForce Server 2020

**Prerequisites**
- FlowForce Server 2020 must be installed (see **Installing FlowForce Server**).
- Perform data migration as a user with administrative (root) privileges.

**To migrate data to FlowForce Server 2020:**

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 installation. This will delete any objects in the new version that were created after installation prior to performing this manual upgrade:
   ```
   sudo rm -rf /var/Altova/FlowForceServer2020/data
   ```

4. Run the `migratedb` command:
   ```
   sudo /usr/local/Altova/FlowForceServer2020/bin/FlowForceServer migratedb
   --olddatadir=/var/Altova/FlowForceServer2017/data --
   datadir=/var/Altova/FlowForceServer2020/data
   ```

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

   ```
   sudo launchctl load /Library/LaunchDaemons/com.altova.FlowForceWebServer.plist
   ```
2.4.3 Uninstalling FlowForce Server

To uninstall FlowForce Server:

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
```

To check whether a service has been stopped, open the Activity Monitor terminal and make sure that the service is not in the list. Double-check that the Activity Monitor shows all processes, not just "My processes" (On the View menu, click All processes.)

3. In Applications, right-click the FlowForce Server icon and select Move to Trash. Note that your FlowForce Server installation directory and configuration data are not removed by this step.

4. Optionally, if you want to remove the FlowForce Server installation directory as well, run the following command:

```
sudo rm -rf /usr/local/Altova/FlowForceServer2020/
```

Note: The steps above do not remove the FlowForce Server configuration data from /var/Altova/FlowForceServer2020. Be aware that once the FlowForce Server configuration data is removed, it cannot be recovered later.

2.4.4 Installation Directory

On macOS, the location of the FlowForce Server installation directory is as follows:

```
/usr/local/Altova/FlowForceServer2020
```
2.5 Windows

2.5.1 Installing FlowForce Server

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

Prerequisites
- Perform installation as a user with administrative privileges.

Installing FlowForce Server

1. Download the installation file from the Altova website (https://www.altova.com/download#server) and run it.
2. Optionally, select the installation language from the box in the lower left area of the wizard, and click Next. The language you select here also determines the language of the FlowForce Server user interface in the Web browser. If necessary, you can change the language later from the command line (see Command Line Interface).
3. Do one of the following:
   a. If you haven't installed Altova LicenseServer yet, leave the default settings as is. The wizard will install the latest version of Altova LicenseServer on the computer where you are running the wizard.
   b. If you haven't installed Altova LicenseServer yet and want to install Altova LicenseServer on another computer, clear the Install Altova LicenseServer on this machine check box, and then choose Register Later. In this case, you will need to install Altova LicenseServer and register FlowForce Server separately.
   c. If Altova LicenseServer has already been installed on your computer but it has 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 with the one indicated on the dialog box. Note that the existing registration and licensing information will be preserved after the upgrade.
   d. If Altova LicenseServer has already been installed on your computer or network, and it has the same version as the one indicated by the wizard, do the following:
      i. Clear the Install Altova LicenseServer on this machine check box.
      ii. Under Register this product with, choose the Altova LicenseServer instance on which you want to register FlowForce Server, or 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.
4. Click Next.
5. Optionally, select any additional Altova server products that you want to install.
6. Optionally, define the network settings and Windows service configuration applicable to FlowForce Server and FlowForce Web Server. For more information, see Network Configuration (Windows Only) and Windows Service Configuration. From the same dialog box, you can optionally define the network and Windows service settings for other Altova products that are being installed at the same time with FlowForce, by clicking their respective tab. If you want to define these settings after installation, click Next.
7. Optionally, set up the FlowForce cluster configuration options. This installation step is applicable for FlowForce Server Advanced Edition only, see Cluster Installation Options on Windows. If you don't need to run FlowForce Server in a cluster, select Standalone.

8. Follow the wizard instructions to complete the installation. When the installation completes, the Altova ServiceController icon is added in the system notification area.

**Licensing FlowForce Server**

1. Ensure that the latest version of Altova LicenseServer (https://www.altova.com/licenseserver) is installed and running either on the local machine or network.

2. If you haven't done so already during installation, register FlowForce Server with Altova LicenseServer. For this, you need to know either the host name or the IP address of the computer where LicenseServer runs. For example, if LicenseServer runs on 127.0.0.1, you can register it as follows:

   ```
   FlowForceServer.exe licenseserver 127.0.0.1
   ```

3. Log on to LicenseServer administration interface and assign the license to the machine where FlowForce Server runs.

For more information, refer to Altova LicenseServer documentation (https://www.altova.com/documentation).

**Post-installation tasks**

1. If you haven't done so already during installation, configure the network settings (see Defining the Network Settings).
2. Log on to the Web administration interface (by default, \texttt{http://localhost:8082}, unless you changed the address and port) and change the default FlowForce Server root password. The default login name and password is root and root, respectively.

2.5.2 \textbf{Windows Service Configuration}

During the installation of FlowForce Server on a Windows operating system, it is possible to configure the following settings:

- How should FlowForce Server start up as a Windows service: automatic, on demand, delayed automatic, or disabled.
- The operating system user account to be used by FlowForce Server as a Windows service: \texttt{Local System}, \texttt{Local Service}, \texttt{Network Service}, or \texttt{Other User}.
- If FlowForce Server is configured to run as \texttt{Other User}, you can set the username and password of this user, similar to how this is done in the Windows Services management console.

\textbf{Note:} The settings listed above can also be configured for FlowForce Web Server.

The settings have the same meaning as when you change the service properties from the Windows Services management console.

To define the settings above during installation, click the relevant tab (FlowForce Server or FlowForce Web Server) on the installation dialog box, and, next to Service configuration, click Change.

If you need to change these settings after installation, open the Windows Services management console, and
then change the properties of the required service from there. To open the Windows Services management console, open a command line window and type Services.msc.

### 2.5.3 Network Configuration

During the installation of FlowForce Server on a Windows operating system, it is possible to configure the following settings:

- Whether unencrypted connections to FlowForce Server are allowed, and, if yes, on which port.

![Webserver Configuration](image)

- Whether secure (SSL-encrypted) connections to FlowForce Server are allowed, and, if yes, on which port. By default, secure connections are disabled. Note that, in order to enable SSL-encrypted connections, the paths to a certificate and a private key file must also be specified. The host (bind address) can be specified from the Setup page after installation (see Defining the Network Settings).
Note: The settings listed above can also be configured for FlowForce Web Server.

To define all of the settings above during installation, click the relevant tab (FlowForce Server or FlowForce Web Server) on the installation dialog box, and then click Change.
If you prefer to configure the settings above at a later time, click **Next**. In this case, FlowForce Server will be installed with the default options as shown under each respective tab of the dialog box. After installation, you can change the settings as described in **Defining the Network Settings**.

### 2.5.4 Migrating Data Manually

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.

**To migrate configuration data manually:**

1. Ensure that Altova ServiceController is running in the system notification area. Otherwise, start the Altova ServiceController.
2. **Stop** the FlowForce Server service and the FlowForce Web Server service.
3. Delete the FlowForce Server data folder installed by the 2020 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. Make sure to specify the correct path to the FlowForce Server .exe file and the configuration data folders applicable to your Windows version (see **How FlowForce Server Stores Configuration Data**). For example, for FlowForce Server 32-bit running on 64-bit Windows 8, the command is:

   ```
   "C:\Program Files(x86)\Altova\FlowForceServer2020\bin\FlowForceServer.exe"
migratedb
   --datadir=C:\ProgramData\Altova\FlowForceServer2020\data --olddatadir=C:\ProgramData\Altova\FlowForceServer2017\data
   ```


### 2.5.5 Installation Folder

On Windows, the default installation path of FlowForce Server is as follows.

<table>
<thead>
<tr>
<th>Windows 7, 8, 10</th>
<th>C:\Program Files\Altova\</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bit Version on 64-bit OS</td>
<td>C:\Program Files (x86)\Altova\</td>
</tr>
</tbody>
</table>
2.6 Migrating Altova Servers

This topic provides instructions for migrating Altova servers from one server machine to another (including across any of the various supported platforms). These instructions use the term "old server" or "old machine" to refer to the machine from which you are migrating data, and the term "new server" or "new machine" to refer to the machine to which you are migrating data.

Altova LicenseServer

To migrate Altova LicenseServer from one server machine to another with the least possible downtime, follow the steps below:

1. Install Altova LicenseServer on the new machine.
2. Stop the Altova LicenseServer service on the new machine and copy the `licenseserver.db` database file from the old machine to the new machine. Make sure to back up the `licenseserver.db` file on the new server in case you want to revert to it for any reason. The `licenseserver.db` file can be found in the LicenseServer application directory and its path will vary depending on the operating system:
   - Windows: `C:\ProgramData\Altova\LicenseServer`
   - Linux: `/var/opt/Altova/LicenseServer`
   - macOS: `/var/Altova/LicenseServer`
3. Stop the Altova LicenseServer on the old machine and start the Altova LicenseServer service on the new machine. This ensures that the two instances of LicenseServer are not running simultaneously.
4. Register all Altova server products with the new Altova LicenseServer. If the number of licenses to be migrated is small, you can do this manually. If you have a large number of licenses to migrate, it is possible to automate registration with the help of custom scripts or code, as follows:
   - Write a command-line script which calls the "register" command for each server product that is under Altova LicenseServer control.
   - Write a program that can open the `*.licsvr` file of each Altova server product programmatically and modify the "host" attribute so that it points to the new Altova LicenseServer. You can find the `*.licsvr` file in the application directory of the respective server product (the path varies by operating system, as shown in the table above).

Altova FlowForce Server

Preparing for migration:

1. Ensure you have administrative privileges both on the old and the new server machine.
2. It is recommended that all jobs should use globally (not locally) stored credentials. If you are using locally stored (inline) credentials in jobs, all such jobs will have to be edited on the new server machine to match the user credentials linked to that operating system. By contrast, if you are using globally stored (standalone) credentials, you will only need to edit the standalone credentials on the new server machine. For more information, see Credentials.
3. If your FlowForce Server uses Windows Active Directory integration, then, for a smooth migration, ensure that the new server machine has the same operating system users and configuration as the old machine.
4. Delete any FlowForce files (jobs, MapForce mappings, StyleVision transformations) that are not in use. In addition to this, perform a general FlowForce Server cleanup and maintenance by running the...
Migrating Altova Servers

Installation

45

built-in functions "archive-log", "cleanup-files", and "truncate-log". The size of the internal FlowForce database can be reduced by running the compactdb command. This will help you significantly reduce the time required to copy the configuration data from the old server to the new server (as described in the steps below).

Performing the migration:

1. On the new server machine, install FlowForce Server. To minimize server downtime during migration, at this time you may also want to register FlowForce Server with Altova LicenseServer, without transferring yet the license to the new server machine; this will be done in a subsequent step. For step-by-step instructions on how to install FlowForce Server, see Installation on Linux, macOS, and Windows.

If you are migrating MapForce Server, StyleVision Server, and RaptorXML(XBRL) Server as well, it is recommended to install these products as part of the FlowForce Server installation.

2. On the old server machine, ensure that there are no running jobs, and stop both the FlowForce Server and the FlowForce Web Server services. For instructions on how to start/stop services on each operating system, see:

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

3. On the old server, locate the data directory of FlowForce Server (see FlowForce Server Application Data), and then copy it to the new server machine, with a temporary name, for example "C:\transfer\migrated_data". The data directory includes any deployed jobs, MapForce mappings and StyleVision transformations, as well as FlowForce users, system functions, settings, and logs (for a complete list, see FlowForce Server Application Data). The duration of this operation will depend on the amount of you FlowForce configuration data.

   The copying time will be greatly reduced provided that you cleaned up the FlowForce Server logs in advance (as explained above, see "Preparing for migration").

4. On the new server machine, stop both the FlowForce Server and FlowForce Web Server services, and then rename the data directory to data_backup or a similar name. This will serve as a backup of the initial FlowForce Server configuration on the new server, in case you will want to revert to it for any reason.

5. From the command-line interface, change the current directory to the FlowForce program directory, and then run the "migratedb" command, for example:

```bash
FlowForceServer migratedb --datadir=C:\ProgramData\Altova\FlowForceServer2020\data --olddatadir=C:\transfer\migrated_data
```

   On UNIX systems, use lowercase flowforceserver to call the executable, and adjust the paths accordingly (see FlowForce Server Application Data). You may also need to prepend
the "./" characters to flowforceserver, to indicate the current directory.
- If the source data directory is of a major version of FlowForce Server that is older than the FlowForce Server version to which you are migrating, the configuration data will be upgraded automatically to the new FlowForce Server version. The same result can be achieved by running manually the "upgradedb" command from the command-line interface.

7. If you haven't done that already, register FlowForce Server with Altova LicenseServer.
8. From the Altova LicenseServer web interface, deactivate your FlowForce Server license from the old server and re-assign it to the new server machine.

Altova MobileTogether Server
Use the Backup and Restore functionality, as described in the MobileTogether Server documentation.

Altova MapForce Server
For MapForce Server, migration to a new machine consists of re-assigning the license from the old machine to the new machine, as follows:

1. Install MapForce Server on the new machine. If it has already been installed as part of FlowForce Server installation, ignore this step.
2. On the new machine, register MapForce Server with Altova LicenseServer.
3. On the old machine, make sure no clients are using the server (for example, no mappings are running).
4. Open the Altova LicenseServer administration page, and deactivate the license from the old MapForce Server machine and re-assign it to the new machine.

Altova RaptorXML(XBRL) Server
For RaptorXML(XBRL) Server, migration to a new machine consists of re-assigning the license from the old machine to the new machine, as follows:

1. Install RaptorXML(XBRL) Server on the new machine. If it has already been installed as part of FlowForce Server installation, ignore this step.
2. On the new machine, register RaptorXML(XBRL) Server with Altova LicenseServer.
3. On the old machine, make sure no clients are using the server (for example, no transformations are running).
4. Open the Altova LicenseServer administration page, and deactivate the license from the old RaptorXML(XBRL) Server machine and re-assign it to the new machine.

Altova StyleVision Server
For StyleVision Server, migration to a new machine consists of re-assigning the license from the old machine to the new machine, as follows:

1. Install StyleVision Server on the new machine. If it has already been installed as part of FlowForce Server installation, ignore this step.
2. On the new machine, register StyleVision Server with Altova LicenseServer.
3. On the old machine, make sure no clients are using the server (for example, no transformations are running).

4. Open the Altova LicenseServer administration page, and deactivate the license from the old StyleVision Server machine and re-assign it to the new machine.
3 Configuring the Server

This section includes FlowForce Server configuration instructions.

You can change the FlowForce Server settings either from the Web administration interface or by editing the FlowForce Server .ini files. Some server maintenance or configuration tasks are also available from the command line interface (see Command Line Interface).

After editing the .ini files manually, you might need to restart the FlowForce services for the settings to take effect. If you need to restart services manually, see:

- Starting and Stopping Services (Linux)
- Starting and Stopping Services (macOS)
- Starting and Stopping Services (Windows)
3.1 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 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/FlowForceServer2020</td>
</tr>
<tr>
<td>macOS</td>
<td>/var/Altova/FlowForceServer2020</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\ProgramData\Altova\FlowForceServer2020</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).</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Do not confuse this file with the flowforceweb.ini file stored in the data directory (described below).</td>
</tr>
<tr>
<td>flowforceweb.html</td>
<td>The setup page is used to register FlowForce Server with Altova LicenseServer and configure the default Web administration interface address and port (see also Opening the Setup Page). This page is regenerated when you restart the FlowForce Web Server service. If the setup page is missing after a fresh installation of FlowForce, start the FlowForce Web Server service to generate it.</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/flowforce.log</td>
<td>These files store the log of messages from FlowForce Web Server and FlowForce Server, respectively. This is applicable to Windows, macOS, as well as Ubuntu 14 and CentOS 6 systems.</td>
</tr>
<tr>
<td>Directory</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>data/flowforce.db</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>data/flowforce.ini</td>
<td>This configuration file defines the port and listening interfaces of FlowForce Server. See also Configuration File Reference.</td>
</tr>
<tr>
<td>data/flowforceweb.ini</td>
<td>This configuration file defines the port and listening interfaces of FlowForce Web Server. See also Configuration File Reference.</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 Setting Environment Variables).</td>
</tr>
</tbody>
</table>
3.2 Opening the Setup Page

The FlowForce Server setup page allows you to specify various network settings, including the host and port on which FlowForce Server and FlowForce Web Server should listen. The setup page is regenerated every time you start the FlowForce Web Server service.

Prerequisites

- The FlowForce Web Server service must be started if it is not running already (see how to start it on Linux, macOS, and Windows). Make sure the random port address on which FlowForce Web Server was started is not blocked by your firewall.

To open the setup page (Linux):

Do one of the following:

- If the Linux machine has a GUI browser, run the flowforceweb.html file in your browser, for example:
  
  sudo firefox file:///var/opt/Altova/FlowForceServer2020/flowforceweb.html

- Open a browser from another machine on the network, and access the URL of the setup page from this browser. You can find the default URL in the flowforceweb.html file located in the application's configuration folder (the URL has the format http://127.0.0.1:34597/setup?key=52239315203).

To open the setup page (macOS):

Do one of the following:

- In Applications, double-click the FlowForce Server icon (note this action requires administrative privileges). The FlowForce Server setup page opens in the Web browser.
- Open the URL of the setup page in the browser. You can find the default URL in the flowforceweb.html file located in the application's configuration folder (the URL has the format http://127.0.0.1:34597/setup?key=52239315203).

If the name of the Mac machine cannot resolve to an IP address (see 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:

a. Click the link mentioned in the message.
b. In the browser's address bar, replace <hostname> with either localhost or the IP address of your Mac.

To open the setup page (Windows):

Do one of the following:
• Click the ServiceController icon ( ) in the system notification area, and then select Altova FlowForce Web > Setup.

• Open the URL of the setup page in the browser. You can find the default URL in the flowforceweb.html file located in the application's configuration folder (the URL has the format http://127.0.0.1:34597/setup?key=52239315203).
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 (see Opening the Setup Page). Alternatively, most of these settings can also 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. (To understand the difference between "FlowForce Web Server" and "FlowForce Server", see How It Works.)

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 enables you to set the host (bind address) and port for the FlowForce Web Server, and specify whether it accepts unencrypted connections, SSL-encrypted connections, or both.
FlowForce Web Server

Unencrypted Connection

- **Enabled:**
  - Checkbox:
  - Options:
    - Local only (127.0.0.1)
    - Other

- **Bind address:**
  - Options:
    - Local only (127.0.0.1)
    - Other

- **Port:**
  - Value: 8082

SSL Encrypted Connection

- **Enabled:**
  - Checkbox:

- **Bind address:**
  - Options:
    - All interfaces (0.0.0.0)
    - Other

- **Certificate file:**
  - Value: C:\secure\flowforceweb.crt

- **Private Key file:**
  - Value: C:\secure\flowforceweb.key

- **Certificate Chain file:**
  - Value: C:\secure\intermediate.pem

By default, plain HTTP (unencrypted) connections to FlowForce Web Server are enabled, unless you modified this during installation* or from configuration files, see Configuration File Reference. On Windows, the Web 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.

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

For information about how to enable SSL-encrypted connections, see Enabling SSL for FlowForce Web Server.

FlowForce Server

The default setting for FlowForce Server accepts only requests from the same machine (127.0.0.1) on port 4646, through an unencrypted connection. If you intend to start jobs as Web services via HTTP from remote machines, select "All interfaces (0.0.0.0)" from the Bind address combo box.
Defining the Network Settings

For information about how to enable SSL-encrypted connections, see Enabling SSL for FlowForce Server.

### Master Instance Encrypted Connection

The settings below must be configured if FlowForce Server is a master instance in a cluster of multiple machines running FlowForce Server, see Distributed Execution and Load Balancing. These settings are also available in the `flowforce.ini` configuration file, see Configuration File Reference.

#### Configuration File Reference

The network settings of both FlowForce Server and FlowForce Web Server can be configured either from the Setup page (see Defining the Network Settings) 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:

**Linux**

```
/var/opt/Altova/FlowForceServer2020/data/flowforce.ini
/var/opt/Altova/FlowForceServer2020/data/flowforceweb.ini
```
Configuring the Server

Defining the Network Settings

| macOS         | /var/Altova/FlowForceServer2020/data/flowforce.ini  
|              | /var/Altova/FlowForceServer2020/data/flowforceweb.ini |
| Windows      | C:\ProgramData\Altova\FlowForceServer2020\data\flowforce.ini  
|              | C:\ProgramData\Altova\FlowForceServer2020\data\flowforceweb.ini |

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:

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

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

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

[Master]
host=0.0.0.0
port=4645
active=1
```

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

```
[Listen]
host=0.0.0.0
port=8082
active=1

[ListenSSL]
active=1
```
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.

<table>
<thead>
<tr>
<th>ssl</th>
<th>(Optional) Enables SSL support for this [Listen] section. Valid values:</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 host 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>
<tbody>
<tr>
<td>port</td>
<td>Specifies the port on which FlowForce (Web) Server will listen.</td>
</tr>
</tbody>
</table>

[SSL]
This section defines the SSL/HTTPS connection settings.

<table>
<thead>
<tr>
<th>certificate</th>
<th>Specifies the absolute path to the certificate file in PEM format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>private_key</td>
<td>Specifies the absolute path to the private key file.</td>
</tr>
</tbody>
</table>
Configuring the Server  
Defining the Network Settings

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.

<table>
<thead>
<tr>
<th>ssl</th>
<th>(Optional) Enables SSL support for the connection between FlowForce Web Server and FlowForce Server. Valid values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>disabled</td>
</tr>
<tr>
<td>1</td>
<td>enabled</td>
</tr>
</tbody>
</table>

| host      | Specifies the host address of FlowForce Server. If SSL is enabled, this value must be the same as the one in the "[ListenSSL]" section of the `flowforce.ini` file. Also, it must correspond to a server name (or IP address, if applicable) that matches the Common Name of the SSL certificate available for FlowForce Server. |

| port      | Specifies the port on which FlowForce Server will listen.                                                                 |

| 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).

| timezone | Specifies the default time zone of FlowForce Web Server, for example `timezone=Europe/Berlin` |

[Master]  
This section is applicable only for the `flowforce.ini` file. It is relevant when multiple FlowForce Server instances run in a cluster, and the current instance is the master instance, see Distributed Execution and Load Balancing.

<table>
<thead>
<tr>
<th>active</th>
<th>Enables encrypted connection to this master instance. Valid values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>disabled</td>
</tr>
<tr>
<td>1</td>
<td>enabled</td>
</tr>
<tr>
<td>binding address</td>
<td>Specifies the binding address of the master FlowForce Server instance. Use 0.0.0.0 to listen on all interfaces.</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>port</td>
<td>The port on which this master instance listens for requests from worker instances.</td>
</tr>
</tbody>
</table>

worker

| worker instances |  |
### 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.)

If you are using FlowForce for exchanging AS2 data, you can also optionally use SSL certificates to sign or encrypt data as part of the AS2 service, see AS2 Integration.

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.

**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
```

**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.
- 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
```

**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.
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, as shown below:

1. Using a text editor such as Notepad, create a new text file (let's call it `intermediate.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 `intermediate.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 intermediate certificate) ...
   --END CERTIFICATE--
   --BEGIN CERTIFICATE--
   ... (primary intermediate certificate) ...
   --END CERTIFICATE
   ```

3. Save the `intermediate.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 may not be suitable for use in production. Your organization will likely have specific security policies concerning SSL certificates and might 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 (https://www.openssl.org/) 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):

```
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:

```
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").

```
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.

```
openssl x509 -req -days 3650 -in privkey.pem -signkey root.key -out root.crt
```
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.
- 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"
subjectKeyIdentifier = 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.
   
   ```
   sudo cp /home/downloads/server_cert.crt /usr/local/share/ca-certificates/
   ```

2. Update the certificate store as follows:
   
   ```
   sudo update-ca-certificates
   ```

On CentOS, follow the steps below:

1. Install the ca-certificates package:
   
   ```
   yum install ca-certificates
   ```

2. Enable the dynamic certificate authority configuration feature:
3. Copy the server certificate to the following directory:

```
cp server_cert.crt /etc/pki/ca-trust/source/anchors/
```

4. Use the command:

```
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.

```
Download Certificate
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 Examine CA certificate
```

**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.
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.

   Perform the following steps only if you are sure of the authenticity of the Web server certificate.

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 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.
- 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 (see Opening the 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), and enter the host name and port where FlowForce Web Server should listen for SSL encrypted connections. 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, this IP address must correspond to the SSL certificate's Common Name.

   c. Enter the path to the certificate and private key file in their respective text boxes. The certificate must be in PEM format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
   d. If applicable, enter the path to the intermediary certificate file (see Preparing Intermediary Certificates).
   e. 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.
Setting up SSL Encryption

Configuring the Server

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. If the SSL settings are not configured correctly or there is a certificate problem, you may no longer be able to access the Setup page. In this case, either use the unencrypted URL (if this is enabled), or change the SSL settings manually (see **Configuration File Reference**).

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**).

### 3.4.4 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 with FlowForce to client machines. 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](#).
Configuring the Server

Setting up SSL Encryption

Certificates with a Certificate Authority or Creating Self-Signed SSL Certificates.

- 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 (see Opening the 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), and enter the host name and port where FlowForce Server listens for SSL encrypted connections. 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, this IP address must correspond to the SSL certificate's Common Name.

   c. Enter the path to the certificate and private key file in their respective text boxes. The certificate must be in PEM format. The file extension of PEM files is usually .pem but it can also be .key, .cert, .cer, or .crt.
   d. If applicable, enter the path to the intermediary certificate file (see Preparing Intermediary Certificates).
   e. Optionally, clear the Enabled check box under "Unencrypted Connection". Note that this will make FlowForce Server unavailable through plain HTTP, so you should take this step only after the SSL encrypted connection works.

<table>
<thead>
<tr>
<th>FlowForce Server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unencrypted Connection</strong></td>
</tr>
<tr>
<td>Enabled:</td>
</tr>
<tr>
<td>Bind address:</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

   | **SSL Encrypted Connection** |
   | Enabled: | ✔ |
   | Bind address: | All interfaces (0.0.0.0) |
   | Certificate file: | C:\secure\flowforce.crt |
   | Private Key file: | C:\secure\flowforce.key |
   | Certificate Chain file: | 

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](#)).

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

You can enable SSL between FlowForce Web Server and FlowForce Server, by editing the .ini files of FlowForce Web Server and FlowForce Server, respectively. You can find both files in the **data** subdirectory of the [FlowForce Server Application Data](#) directory.

**Do the following in the flowforce.ini file:**

1. In the [ListenSSL] section, type values for the following parameters: **ssl**, **active**, **host**, **port**, for example:

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

   - The **ssl** and **active** parameters must be set to 1 (enabled).
   - The **host** must correspond to a server name (or IP address, if applicable) that matches 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_chain=
   certificate=C:\secure\flowforce.crt
   private_key=C:\secure\flowforce.key
   ``

**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 three parameters: **ssl**, **host**, and **port**, for example:
Configuring the Server

Setting up SSL Encryption

```
[FlowForce]
ssl=1
host=server.my.domain.com
port=4647
```

- The `ssl` parameter must be enabled (set to 1) for SSL connections to be enabled.
- The `host` must be the same as the one in the "[ListenSSL]" section of the `flowforce.ini` file.
- The `host` must correspond to a server name (or IP address, if applicable) that matches the Common Name of the SSL certificate available for FlowForce Server.
- The `port` must be different from the default 4646 port used for unencrypted connections. Typically, this value is 4647.

**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. Click Administration.
2. Click Settings.
3. Under Input format, select the default time zone.
4. 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. Click **Administration**.
2. Click **Settings**.
3. 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.
4. If your SMTP server requires authentication, click the plus icon next to **User authentication** and enter the username and password.
5. 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 Changing the 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, click Administration, and then click 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 it 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:
  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 Changing the 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.

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 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.

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 change the default size of log entries, go to Administration | Settings and change the parameters grouped under "Logging limits". 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>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default string value logging limit</td>
<td>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.</td>
</tr>
<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>
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:

```
[CentOS 6]:   sudo initctl start flowforcewebserver
[CentOS 7]:   sudo systemctl start flowforcewebserver
[Debian 8]:   sudo systemctl start flowforcewebserver
[RedHat]:     sudo initctl start flowforcewebserver
[Ubuntu 14.04]:   sudo initctl start flowforcewebserver
[Ubuntu 16.04]:   sudo systemctl start flowforcewebserver
```

To stop the FlowForce Web Server service:

```
[CentOS 6]:   sudo initctl stop flowforcewebserver
[CentOS 7]:   sudo systemctl stop flowforcewebserver
[Debian 8]:   sudo systemctl stop flowforcewebserver
[RedHat]:     sudo initctl stop flowforcewebserver
[Ubuntu 14.04]:   sudo initctl stop flowforcewebserver
[Ubuntu 16.04]:   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:

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

To start the FlowForce Web Server service:

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

To stop the FlowForce Server service:

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

To stop the FlowForce Web Server service:

```
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 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>\FlowForceServer2020\bin` folder. Your DLL file will have a name of the form `FlowForceServer2020_lc.dll`. The `_lc` part of the name contains the language code. For example, in `FlowForceServer2020_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.
3.13 Setting Environment Variables

When MapForce Server mappings or StyleVision Server stylesheets run under FlowForce Server management (as jobs), they may require environment variables to be set (for example, the CLASSPATH, in order 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 application data directory:

Table 1: FlowForce Server application data directory

<table>
<thead>
<tr>
<th>Platform</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>/var/opt/Altova/FlowForceServer2020/data/tools/</td>
</tr>
<tr>
<td>macOS</td>
<td>/var/Altova/FlowForceServer2020/data/tools/</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\ProgramData\Altova\FlowForceServer2020\data\tools\</td>
</tr>
</tbody>
</table>

If the .tool file does not exist in the directory above, copy it from the installation directory of FlowForce Server. You will find the .tool file in the tools directory relative to the FlowForce Server installation directory (it is assumed that MapForce Server or StyleVision Server were installed after FlowForce Server):

Table 2: FlowForce Server installation directory

<table>
<thead>
<tr>
<th>Platform</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>/opt/Altova/FlowForceServer2020/</td>
</tr>
<tr>
<td>macOS</td>
<td>/usr/local/Altova/FlowForceServer2020/</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\Program Files\Altova\FlowForceServer2020\</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files (x86)\Altova\FlowForceServer2020\</td>
</tr>
</tbody>
</table>

If the .tool file exists neither in the FlowForce application data directory nor in the FlowForce installation directory, 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.

Make sure to copy the .tool file to the FlowForce Server application data directory (NOT the installation directory) before editing it. The .tool file in the FlowForce Server application data directory above takes precedence over the .tool file existing in other directories.

Warning: If you run the migratedb command during upgrade 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, ensure that you keep in the application data directory only the .tool files that you actually need.

You can add the required environment variables under the [Environment] section within the .tool file. For example, a .tool file which sets the CLASSPATH variable looks as follows:

```
[Environment]
CLASSPATH=:/usr/local/jdbc/oracle/ojdbc6.jar
```
Sample .tool file (Linux)

Environment variables set in the .tool file override environment variables defined by other means.

For information about executing shell commands or scripts as FlowForce Server jobs, see the `/system/shell/commandline` function.
4 Distributed Execution

To improve data throughput and provide basic fault tolerance, you can configure multiple FlowForce Server instances to run as a cluster. This provides the following benefits:

- Load balancing
- Leaner resource management
- Scheduled maintenance
- Reduced risk of service interruption

Load balancing
When hardware limits cause FlowForce Server to be overwhelmed by multiple job instances running simultaneously, it is possible to redistribute workload to another running instance of FlowForce Server (a so-called "worker"). You can set up a cluster comprised of a "master" machine and multiple "worker" machines and thus take advantage of all the licensed cores in the cluster.

Leaner resource management
One of the machines designated as a "master" continuously monitors job triggers and allocates queued items to workers, or even to itself, depending on configuration. You can control the queue settings and decide, for each job, the queue where it should be assigned. For example, you can optionally configure the master machine not to process any job instances at all and thus free up its resources and dedicate them exclusively to continuous provision of FlowForce Service as opposed to data processing.

Scheduled maintenance of workers
You can restart or temporarily shut down gracefully any running instance of FlowForce Server that is not the "master", without interrupting provision of service. Note that the "master" is expected to be available at all times; restarting or shutting it down will still interrupt provision of service.

Reduced risk of service interruption
In case of disasters such as hardware failures, power outages, unplugged network cables, and similar, the impact depends on whether the affected machine is a "worker" or a "master":

- If the machine is a "worker", any running FlowForce job instances on that worker will be lost. However, general provision of FlowForce service will not be lost, because new instances of the same job will be taken over by a different worker (or by the master, if configured). The execution status of the job, including failure, is reported to the master and visible in the job log, so that an administrator can take appropriate action manually.
- If the machine is a "master", provision of service is lost. In this case, new job instances cannot start as long as the master is unavailable.
4.1 Distributed Execution Terminology

The following terminology is used in conjunction with distributed execution and load balancing.

Server Instance
A server instance is a running and licensed installation of FlowForce Server. Both services (FlowForce Web Server and FlowForce Server) are assumed to be up and running on the machine.

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. Job instances are distributed within the cluster as defined by the execution queue associated with the job. A job instance will always run in its entirety on a single cluster member.

Cluster
A cluster represents several service instances of FlowForce Server that communicate for the purpose of executing jobs in parallel or redistributing jobs if any instance is not available. A cluster consists of one "master" FlowForce Server and one or several "workers".

Master
A "master" is a FlowForce Server instance that continuously evaluates job-triggering conditions and provides the FlowForce service interface. A master is aware of worker machines in the same cluster and may be configured to assign job instances to them, in addition to (or instead of) processing job instances itself.

Worker
A FlowForce Server instance that is configured to communicate with a master instance instead of executing any local jobs. A worker can execute only jobs that a master FlowForce Server has assigned to it.

Execution Queue
An execution queue is a "processor" of jobs; it controls how job instances run. In order to run, every job is assigned to a target execution queue. You can assign a job to an execution queue while configuring the job, and it will be submitted to that execution queue at runtime. The queue controls how many job instances (of all the jobs assigned to the queue) can be running at any one time, the delay between runs, and other settings. Queues can be local to the job, or shared by multiple jobs. When multiple jobs are assigned to the same execution queue, they will share that queue for executing.
4.2 Operating FlowForce Server in "Master" Mode

A "master" is a FlowForce Server instance that continuously evaluates job-triggering conditions and provides the FlowForce service interface. A master is aware of worker machines in the same cluster and may be configured to assign job instances to them, in addition to (or instead of) processing job instances itself.

Immediately after installation, the FlowForce Server instance acts as the master of a one-machine cluster (which includes itself). However, work will not yet be distributed, since there are no workers to take over the workload. To set up a cluster, install additional FlowForce Server instances and convert them to "worker" mode, as shown further in this documentation. A cluster ready for load balancing is assumed to be set up when at least one machine acts as worker, in addition to the master machine.

Note: Only one master machine can exist in a cluster; the number of workers is not limited.

There is no difference between operating a standalone FlowForce Server instance compared to a master instance. You configure jobs and view the processing log in exactly the same way. The only difference is that a master communicates with workers from the same cluster. In the cluster management page, you can view at all times the list of workers joined to the master, including those that attempted to join but did not confirm the security token. From this page, you can generate security tokens to confirm workers as such, and you can also remove workers completely. For further information, see Converting FlowForce Server to "Worker" Mode and Removing a worker from the master.

The master machine is responsible for continuous provision of service, collecting the status of job instances assigned to workers, and reporting the outcome. For this reason, it is important that the master machine is balanced according to the demands of your processing environment. To achieve that, you can redirect some or all jobs into queues that will be processed by workers, while the master will mainly provide the service interface. The master may also be configured to take some processing workload itself, in the event that no workers are available, see Setting up Distributed Execution.
4.3 Converting FlowForce Server to "Worker" Mode

Converting FlowForce Server to "worker" mode means that you allocate its resources exclusively for processing job instances as requested by a "master" FlowForce Server instance. Once converted to a worker, the FlowForce Server can no longer execute any locally configured triggers and jobs, unless it is converted back to normal mode. The "worker" status of a FlowForce Server instance is displayed in the web administration interface at all times.

You can convert FlowForce Server to worker mode at any time, from the cluster management page, as illustrated below. When worker mode is no longer required, you can terminate it and convert FlowForce Server back to normal mode, see Terminating the "Worker" Mode.

On Windows, it is possible to specify some cluster configuration options during installation, see Cluster Installation Options on Windows.

Prerequisites

- The FlowForce Server instance must be installed, licensed, and running. The same requirement applies to a second FlowForce Server instance, the one that will act as "master".
- On each machine where you need to take cluster-related actions, your FlowForce user account must have the "Maintain cluster" privilege, see How Privileges Work. By default, the root user account has this privilege.
- If the worker will run jobs that require a MapForce Server, StyleVision Server, RaptorXML Server, or RaptorXML+XBRL Server license, these tools must be installed and licensed on the worker instance. If the master instance will not run such jobs (assuming that all jobs and queues are configured to redistribute workload to workers), then these tools need not be installed on the master.
- Open the setup page on the master machine and check that connections to the master instance are enabled, and the bind address and port are set, for example:

![Master Instance Encrypted Connection]

See also Opening the Setup Page.

Converting a running FlowForce Server to "worker" mode

1. Log on to FlowForce Server instance that is to become the worker, see Logging on to FlowForce Server.
2. Access the cluster management interface, by clicking Administration, and then Cluster.
3. Click Request to Join Master Instance.
4. Enter the host name of the machine that is to become the master.
5. Optionally, enter a custom text message to identify your join request (in this example, "Hello from worker machine!").
6. Click **Send Request to Join Master**.

Ensure that the bind address is configured correctly on the master machine and the port is not blocked by the firewall, see the prerequisites above.

7. Log on to the FlowForce Server that is to be the master and access the cluster management interface.
8. Find the join request entry originating from the worker machine and click **Accept Request**.

9. Click **Show Token** next to the request originating from the worker machine. The secret key required to join this worker to the cluster is displayed.

10. Transfer the token to the worker machine. An easy way to do so is to open both FlowForce Server instances in the browser and copy-paste the key from one window/tab to another. Alternatively, you can use email or other means to transfer the key.

11. Access the cluster management interface on the worker machine.
12. Click **Complete to Join Master Instance**.
13. Enter the host name of the master, paste the secret key (token) in the provided text box, and click **Complete to Join Master**.

On success, a notification message is displayed in the page.

FlowForce Server is now in worker mode and can only execute jobs on request by the master machine. If you need to switch the machine back to standalone mode, click **Leave Master Instance**. See also **Terminating the "Worker" Mode**.
4.4 Cluster Installation Options on Windows

When you install FlowForce Server on Windows, one of the wizard steps enables you to define the cluster status of FlowForce Server: standalone, master, or worker.

You can also set up the cluster after installation. Otherwise, select one of the following options:

a. (Default) Select Standalone if you do not plan to run the multiple FlowForce Server instances as a cluster.

b. Select Master if you plan to use this FlowForce Server instance as master instance. This option requires that you specify the port number where the master FlowForce Server instance should listen. Notice this port must be different from port numbers used by of "FlowForce Server" and "FlowForce Web Server" services.

c. Select Worker if you plan to use this FlowForce Server instance as worker instance.

If you selected Master, the port of the "Master Instance Encrypted Connection" in the setup page is set to the value you specified during installation. You can then proceed to adding workers to the cluster. To do this, either install new FlowForce Server instances as workers as shown above, or convert existing FlowForce Server instances to worker mode, see Converting FlowForce Server to "Worker" Mode. Regardless of the approach you choose, note that you will need to confirm manually the security token of each worker before it is joined to the master, as described in Converting FlowForce Server to "Worker" Mode.

If you selected Worker, you will be redirected to the cluster management page after your first login as root user (or as any user that has the "Maintain cluster" privilege). From the cluster management page, you can then
request to join the master and complete the process as described in Converting FlowForce Server to "Worker" Mode.
4.5 Setting up Distributed Execution

At the core of distributed execution lies the concept of execution queues.

An execution queue is a "processor" of jobs; it controls how job instances run. In order to run, every job is assigned to a target execution queue. You can assign a job to an execution queue while configuring the job, and it will be submitted to that execution queue at runtime. The queue controls how many job instances (of all the jobs assigned to the queue) can be running at any one time, the delay between runs, and other settings. Queues can be local to the job, or shared by multiple jobs. When multiple jobs are assigned to the same execution queue, they will share that queue for executing.

Shared queues provide a flexible mechanism to control server load either on a single FlowForce machine, or when multiple FlowForce Server instances run as a cluster. Configuring load balancing is a multi-step process:

1. First, you create a queue from a dedicated page, similar to how you would create other FlowForce configuration data, such as credentials or jobs.
2. For each queue, you define its processing settings. For example, you can configure a queue to run only on master, only on workers, or both. It is also possible to define basic fallback criteria. For instance, a queue may be configured to run by default on master and all its workers; however, if all workers become unavailable, the queue will fall back to master only.
3. Edit the configuration of each job and assign the job into the custom queue created previously.

Creating queues

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.

To create a queue:

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 other settings, see “Queue settings” below.
4. Click Save.

Queue settings
The settings available for configuration in a queue are listed below.

<table>
<thead>
<tr>
<th>Run on</th>
<th>Specifies how all job instances from this queue are to be run:</th>
</tr>
</thead>
<tbody>
<tr>
<td>master or any worker</td>
<td>Job instances that are part of this queue will run indiscriminately on the master or worker machines, depending on available server cores.</td>
</tr>
<tr>
<td>master only</td>
<td>Job instances will run only on the master machine.</td>
</tr>
<tr>
<td>any worker only</td>
<td>Job instances will run on any available worker but never on master.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minimum time between runs</th>
<th>An execution queue provides execution slots, where the number of available slots is governed by the “maximum parallel instances” setting multiplied by the number of workers assigned according to the currently active rule. Each slot will execute job instances sequentially.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The &quot;Minimum time between runs&quot; 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.</td>
</tr>
</tbody>
</table>
**Maximum parallel runs**

This option defines the number of times the same job may be executed in parallel on the server.

Parallel processing depends on the number of server cores available to FlowForce Server. For example, if you set this value to 12 and only two server cores is licensed, 12 jobs will appear as running in parallel; however, only two jobs will make progress at any given time.

You can define multiple sets of queue settings, each with different processing requirements, by clicking the **+** button. To change the priority of a specific set of settings (let's call it "rule"), click the **Move up** or **Move down** buttons. For example, you can define a rule for the case when only master is available, and another rule for the case when both the master and workers are available. This enables you to create a fallback mechanism for the queue, depending on the state of the cluster at a given time. When processing queues, FlowForce Server constantly monitors the state of the cluster and "knows" if any worker is unavailable. So, if you defined multiple queue settings rules, FlowForce Server evaluates them in the defined order, top to bottom, and picks the first rule that has at least one cluster member assigned according to "run".

As an example, let's consider a setup where the cluster includes one master and four worker machines. The queue settings are defined as shown below:

![Queue settings](image)

With the configuration illustrated above, FlowForce would process the queue as follows, depending on the current state of the cluster:

- If all workers are available, the top rule matches and will be applied. Namely, up to 16 job instances are permitted to run simultaneously (4 instances for each worker). The minimum time between runs is 0 seconds.
- If only three workers are available, the top rule still matches. Namely, up to 12 job instances are permitted to run simultaneously, and the minimum time between runs is 0 seconds.
• If no workers are available, the second rule matches and will be applied. Namely, up to 1 instance is permitted to run simultaneously, and the minimum time between runs is 5 seconds.

This kind of configuration makes execution still possible in the absence of workers. Notice that the "master only" rule is a bit more strict (1 instance only, and 5 seconds delay between runs) so as not to take away too much processing power from the master machine when all workers fail.

Assigning jobs into queues

Once you have configured the queue, the next step is to edit the configuration of each job that you want to assign to this queue. You will find the queue settings in the job configuration page, in the "Queue Settings" group:

![Queue settings](image)

**Note:** If you select **Define local queue**, FlowForce Server will assign, at job runtime, instances of this job into a default queue, with the local settings you specify, see also [Defining Queue Settings](#). Local queues do not support distributed processing. The queue must be created standalone (external to the job) in order to benefit from distributed processing.
4.6 Terminating the "Worker" Mode

Whenever you need to convert a worker machine to a standalone FlowForce Server instance, you can do so from the cluster management interface of the worker machine:

1. Make sure that your FlowForce user account has the "Maintain cluster" privilege, see How Privileges Work.
2. On the worker machine, click Administration, and then click Cluster.
3. Click Leave Master Instance.

This converts the FlowForce Server instance to normal operating mode; however, it still remains registered with the master instance until explicitly removed by the master. In this state, you can still generate a secret key for this worker on the master machine in the event that you want to rejoin the cluster. To remove a worker completely from the master machine as well, see the instructions below.

Removing a worker from the master

On the master machine, any workers that requested to join the master instance in the past are visible at all times in the cluster management page. This includes both workers that confirmed their security token and those that have not. The latter category includes machines that were converted to normal (not worker) status.

Removing a worker without first terminating worker mode leaves the worker in worker mode, and it will not be able to connect to the master any longer. To make connection to master possible again, perform the Leave Master Instance action on the worker machine, as described above.

To remove a worker from the master instance:

1. Make sure that your FlowForce user account has the "Maintain cluster" privilege, see How Privileges Work.
2. On the master machine, click Administration, and then click Cluster. The list of workers is visible in the "Members" section.
3. Click Remove Worker. A confirmation message appears.
4. Click Confirm and Remove.
Managing User Access

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

This section includes the following topics:

- Creating Users
- Creating Roles
- Importing Domain Users and Roles
- Default Users and Roles
- Renaming Users and Roles
- Assigning Roles to Users
- Assigning Roles to Other Roles
- Resetting the Root Password

5.1.1 Creating 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 ( _ ), dash ( - ), and full stop ( . )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Password</th>
<th>Enter the user's password.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Re-type password</th>
<th>Re-type the user's password.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Change password on next login</th>
<th>If you select this check box, the user will be prompted to change password on next login.</th>
</tr>
</thead>
</table>

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.
5.1.2 Creating 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.

5.1.3 Importing 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 Allow any domain users to log in setting 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 Allow any domain users to log in check box, and import the domain users as shown below.

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

To import domain user accounts into FlowForce Server:

1. Under Settings, select the Enable check box under “Directory Services” and configure your preferred Directory Service provider as described in Changing the Directory Service Settings.
2. Under Administration, 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, so, 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 on the webpage dialog. In 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". Refer to Microsoft documentation for further information about Ambiguous Name Resolution in Active Directory.

6. Select the check box next to records that are to be imported, and then click **Import Selected**. Be patient while records are being retrieved from the Active Directory. Waiting time increases if the domain is not local.

**To import domain groups into FlowForce Server:**

1. Under **Administration**, click **Roles**.
2. Click **Import Domain Roles**.
3. Follow the steps 4-6 above.

**See also**

- Changing the Directory Service Settings
5.1.4 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 Exposing Jobs as Web Services).</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.

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).

5.1.5 Renaming 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.

### 5.1.6 Assigning 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**.
5.1.7 Assigning 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

5.1.8 Resetting 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.
5.2  Privileges

This section includes the following topics:

- How Privileges Work
- Viewing Privilege Reports

5.2.1  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>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define execution queues</td>
<td>Grants rights to create and maintain job execution queues. This includes both queues local to the job and external queues defined outside of the job. External queues are used in conjunction with distributed execution, see Load Balancing and Distributed Execution.</td>
</tr>
<tr>
<td>Maintain cluster</td>
<td>Grants rights to perform actions that let one manage multiple FlowForce Server instances as a cluster. For example, a user requires this privilege in order to be able to convert the current service instance of FlowForce Server into a &quot;worker&quot;, see Load Balancing and Distributed Execution.</td>
</tr>
<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. |
Managing User Access Privileges

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

**Read users and roles**

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 and roles.

By default, only the user root has this privilege.

**Retrieve sensitive data**

This privilege grants the right to retrieve and view the following categories of sensitive data as plain text:

- Passwords
- Certificate private keys
- OAuth 2.0 access tokens, refresh tokens, and client secrets.

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.

**Set own password**

This privilege grants to 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.

**Stop any job**

This privilege grants the right to stop any running FlowForce Server job, regardless of the user who created it.

**View unfiltered log**

By default, users can see log entries related to configurations to which they have "read" 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.

---

**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 Report

<table>
<thead>
<tr>
<th>Privileges Report (Sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maintain global settings</strong></td>
</tr>
<tr>
<td><strong>Maintain users, roles and privileges</strong></td>
</tr>
<tr>
<td><strong>Override security</strong></td>
</tr>
<tr>
<td><strong>Read users and roles</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Set own password</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Stop any job</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>View unfiltered log</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>User</th>
<th>Privileges</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Althia 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 Mauer</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>Netsuo 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>
5.3 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

5.3.1 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>
<tr>
<td>Note:</td>
<td>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.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user the right to enter the container (more specifically, the container appears to the user as disabled).</td>
</tr>
</tbody>
</table>
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>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherit</td>
<td>Provides to the user the same configuration object–related rights as those defined on the parent container.</td>
</tr>
<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>
</tbody>
</table>
| Read, Write  | Grants the user rights to modify any configuration object within the container (for example, edit the trigger of a job).  
<br>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 view the details of any configuration objects within the container (more specifically, configuration objects appear to the user as disabled). |

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

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherit</td>
<td>Provides to the user the same credential–related rights as those defined on the parent container.</td>
</tr>
<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>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherit</td>
<td>Provides to the user the same queue rights as those defined on the parent container.</td>
</tr>
<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. In addition, if a job exposes an AS2 service, then this permission controls access to the AS2 service exposed by the job, see [Receiving AS2 Messages](#).

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

<table>
<thead>
<tr>
<th>Use</th>
<th>Grants the user rights to access the service and thus execute the job via the request interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notes</strong></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>If you grant Use permission to user anonymous, the service becomes publicly available and does not require authentication.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user the right to access the job as a Web service.</td>
</tr>
</tbody>
</table>

**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".

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same function–related rights as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>Grants the user rights to call (refer to) any function defined inside the container.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user rights to call (refer to) any function defined inside the container.</td>
</tr>
</tbody>
</table>

**Certificate**

This permission defines how a user can access a digital security certificate from the current container. For more information, see Configuring AS2 Certificates.

| Inherit | Provides to the user the same rights as those defined on the parent container. |
### Use
Grants the user rights to use (refer to) any certificate defined inside the container.

### No access
Denies the user rights to use (refer to) any certificate defined inside the container.

### AS2 Partner
This permission defines how a user can access AS2 partner objects defined in the current container. For more information, see Configuring AS2 Partners.

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same rights as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>Grants the user rights to use (refer to) any AS2 partner object defined inside the container.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user rights to use (refer to) any AS2 partner object defined inside the container.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Inherit</th>
<th>Provides to the user the same resource-related rights as those defined on the parent container.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>Grants the user rights to reuse (refer to) any resources defined in this container.</td>
</tr>
<tr>
<td>No access</td>
<td>Denies the user the right to reuse (refer to) any resources defined in this container.</td>
</tr>
</tbody>
</table>

### 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>
5.3.2 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:
|         | • When you deploy mappings from MapForce to FlowForce Server.
|         | • When you deploy transformations from StyleVision to FlowForce Server.
|         | You can, however, deploy mappings or transformations to a different container, if required. |
| /RaptorXML | This container is present if you licensed RaptorXML Server. It stores the validation and other functions specific to RaptorXML Server. |
| /system | The system container contains the FlowForce Server system functions. It is not recommended to make changes to this container. |

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.
Managing User Access Permissions and Containers

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></td>
<td>Credential</td>
</tr>
<tr>
<td></td>
<td>Function (includes built-in functions, MapForce mappings and StyleVision transformations)</td>
</tr>
<tr>
<td></td>
<td>Job</td>
</tr>
<tr>
<td></td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>Certificate, see AS2 Integration.</td>
</tr>
<tr>
<td></td>
<td>Certificate (with private key), see AS2 Integration.</td>
</tr>
<tr>
<td></td>
<td>AS2 Partner (see AS2 Integration)</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>
### Property | Description
--- | ---
Modified by | Specifies the name of the user who modified the object.
Next run | For jobs scheduled to run with time triggers, this column specifies the date and time of the next run, as defined in the job settings.
View log | For jobs, this button provides quick access to the execution log of the corresponding job.

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.

### 5.3.3 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.

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.

5.3.4 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.

For the description of each permission type, see [How Permissions Work](#).

### 5.3.5 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 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.
5.3.6  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.

<table>
<thead>
<tr>
<th>Permissions for /public</th>
</tr>
</thead>
<tbody>
<tr>
<td>User or Role name</td>
</tr>
<tr>
<td>Container:</td>
</tr>
<tr>
<td>Configuration:</td>
</tr>
<tr>
<td>Credential:</td>
</tr>
<tr>
<td>Queue:</td>
</tr>
<tr>
<td>Service:</td>
</tr>
<tr>
<td>Function:</td>
</tr>
<tr>
<td>Certificate:</td>
</tr>
<tr>
<td>AS2 Partner:</td>
</tr>
<tr>
<td>Security:</td>
</tr>
</tbody>
</table>

*Default permissions of the /public container*

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).
5.4 Password Policies

This section includes the following topics:

- How Password Policies Work
- Creating and Assigning Password Policies

5.4.1 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

5.4.2 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**.
6 Configuring Jobs

This section includes procedures and concepts applicable to FlowForce Server job configuration.
## 6.1 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, or OAuth security details.

Credentials can be of the following type:

- **Password** (the combination of a username and password)
- **OAuth 2.0**

Credentials of type **password** 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. **Password** 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.

Credentials of type **OAuth 2.0** are necessary in jobs that call Web services where OAuth 2.0 authentication is required.

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.

**OAuth 2.0** credentials can be defined only as standalone (not local) credentials, and subsequently be referenced from any jobs where they are required.

### 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.
6.1.1 Defining 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>Credential type</td>
<td>Choose a credential type (Password or OAuth 2.0). For more information about the latter, see OAuth 2.0 Credentials.</td>
</tr>
</tbody>
</table>
### Configuring Jobs

#### Credentials

- **User name**: 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 `username@domain`.

  If the credential usage is for HTTP or FTP (see below), this may also be the HTTP or FTP user name.

- **Password**: 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.

- **Allow usage for HTTP**: Select this check box if the credential will be referenced in jobs that call Web services which require basic HTTP authentication.

- **Allow usage for FTP**: Select this check box if the credential will be referenced in jobs that connect to FTP servers using `/system/ftp` functions.

- **Allow usage for job execution**: Select this check box if the credential identifies an operating system user account. In order to run successfully, any job requires a credential with this usage enabled.

  Ensure that the user account identified by the credentials has sufficient rights on the operating system. For example, if credentials are going to be referred in a job that writes to a directory, the user account must have rights to write to that directory.

#### 6.1.2 OAuth 2.0 Credentials

In addition to credentials of type `password`, you can also create credential objects that are OAuth 2.0 authorization details. You can use OAuth 2.0 credentials in FlowForce Server jobs that call Web services where OAuth 2.0 authentication is required.

You can create OAuth credentials in the same way as password credentials, see Defining Credentials. Like with other FlowForce Server objects, users can view or access OAuth credentials only if they have the corresponding permissions, see How Permissions Work.

The fields associated with an OAuth 2.0 credential object are listed below. To obtain these values, you must first register with the Web service provider (for example, Google API Console, Facebook API, Bitbucket API, and so on).

- **Redirect URI**: Specifies the URI where the authorization server will send responses to FlowForce Server (tokens or errors). This field is filled automatically by FlowForce Server.

- **Authorization Endpoint**: Specifies the URI from where FlowForce Server initiates authorization flows. You can obtain this value after registering with the Web service provider.

- **Token Endpoint**: Specifies the URI from where FlowForce Server initiates token flows. You can obtain this value after registering with the Web service provider.
| **Client ID** | The identifier of the client application (FlowForce Server, in this case). You can obtain this value after registering with the Web service provider. |
| **Client secret** | The secret associated with the client application. You can obtain this value after registering with the Web service provider. |
| **Scope** | The scope of the client application, if required by the provider. You can obtain this value after registering with the Web service provider. |
| **Access token** | This is the access token returned by the authorization server. The FlowForce Server job will execute successfully only if the resource server determines that the access token is correct and valid. To obtain this value manually the first time when you create the OAuth credential, fill all the other fields (except **Refresh token**), and then click **Authorize and Save**. This token expires after a period of time set by the Web service provider. If the token has expired, FlowForce Server will request a new one from the authorization server, using the **Refresh token** value. |
| **Refresh token** | This is the refresh token returned by the authorization server. It is required when the **Access token** expires (see above). In rare cases when the access token never expires, this is not necessary. |

### 6.1.3 Referring 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.
- You have created an OAuth 2.0 credential. You may refer this credential in a job that calls a Web service that requires OAuth 2.0 authorization.

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.
6.2 Creating Jobs

This topic provides generic instructions on how to create jobs from the FlowForce Server Web administration interface. The generic instructions are meant to help you understand the structure of jobs and their available settings; however, not every step and setting listed below is applicable in real-life scenarios. Configuring jobs is a flexible process by definition and there might be more than one way to achieve the same result. To help you get an idea of various things you can do with jobs, this documentation also includes miscellaneous job configuration examples.

Prerequisites

- Make sure that you have the following permissions for the container where you are creating the job (see Viewing Container Permissions):
  - Container - Read, Write
  - Configuration - Read, Write

Tips

- 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 (see Defining Credentials).
- To avoid common errors when creating jobs, and especially if you intend to pass values between steps or between jobs, see The FlowForce Expression Language.

To create a job:

1. Click Configuration, and then navigate to the container in which you want to create the job.
2. Click Create, and then 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 (see Managing Input Parameters).
5. Under Execution Steps, add the steps of the job (see Managing Steps). 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 under Execution Result (see Declaring the return type of a job).
7. If you want FlowForce Server to cache the returned result, specify the caching preferences (see Caching Job Results).
8. Under Triggers, add the trigger (or triggers) that will fire the job (see Managing Triggers). If the job will be exposed as a Web service, adding a trigger is not necessary.
9. If you want to expose the job as a Web service, click the Make this job available via HTTP... check box (see Exposing Jobs as Web Services).
10. Under Credentials, select an existing credential record or specify a local credential (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 (see Caching Job Results).
12. Optionally, define the job's queue settings (see Defining Queue Settings).
13. Click Save. FlowForce Server validates the entered information and, if necessary, highlights in red any fields that require your attention.
6.3 Duplicating jobs

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**.

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.

Currently, 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**.
6.4 Managing 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.</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></td>
<td>You should select the data type according to the data type of the expected parameter value.</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>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add parameter.</td>
<td></td>
</tr>
<tr>
<td>Delete parameter</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------</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>
### 6.5 Managing 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.

You can create the following step types.

| 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. |
| 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: |
| 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: |

```plaintext
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.

For more information, see Processing Steps Conditionally.
```

FlowForce executes the step until it finishes looping through all items of the sequence.
sequence expression.

**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).

To add a step, click the button corresponding to the step type.

### Execution Steps

- [new Execution step](#)
- [new Choose step](#)
- [new For-each step](#)
- [new error/success handling step](#)

Use the following buttons to manage steps.

- ![Add step](#)
- ![Delete step](#)
- ![Duplicate step](#)
- ![Move step up or down](#)
- ![Undo a previous delete action](#)
## 6.6 Processing 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 diagram](image)

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
- Handling Step Errors
6.7 Processing 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.

![Abstract representation of a job which includes conditional processing](image)

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 "fiileexists" 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 \texttt{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 \textit{The FlowForce Expression Language}.
6.8 Handling Step Errors

When a step of a job fails, the job is aborted. 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 handling step". 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.

![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** (instructs the system to perform some action if all steps in the protected block were successful)
- **On Error** (instructs the system to perform some action if any of the steps in the protected block were not successful)
- **Always** (instructs the system to perform some action 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 \textbf{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 \textit{Adding Error Handling to a Job}.13

\textbf{Considerations when implementing exception handling}

In the configuration illustrated previously, a single step was being handled, namely, the first step of the job. 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.

\textbf{When multiple steps exists 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.

\textbf{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 \textit{On Success, On Error} or \textit{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 result1, the second step has 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, 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, result2 is accessible from step 3 but it is not accessible from the error handler. The result3 is not accessible at all, because there are no other steps following it.
- Finally, the result of the error handler, 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 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](#).
6.9 Declaring the Return Type of a Job

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 declaring the return type, keep in mind the following:

- Declaring the return type is meaningful only for jobs that actually return a result.
- If you want to cache the result of the 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 similar to "Types string and result do not match". When type matching errors such as this one 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 (see Handling data types in steps).

To declare the return type of the job:

1. Create a new job or open an existing one for editing.
2. Select a return type in the Execution Result section.

The available return types are as follows.

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignore/discard</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>string</td>
<td>Specifies that the data returned by the job represents a text value (string).</td>
</tr>
<tr>
<td>stream</td>
<td>Specifies that the data returned by the job represents a stream.</td>
</tr>
<tr>
<td>number</td>
<td>Specifies that the data returned by the job is a numeric value.</td>
</tr>
<tr>
<td>boolean</td>
<td>Specifies that the data returned by the job is a Boolean value.</td>
</tr>
</tbody>
</table>
6.10 Handling 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:

   `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:

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 Set to, 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.
Referring to Windows Network Paths

As you create jobs, you will typically need to refer to file paths on the machine where FlowForce Server runs, or file paths on the network. When referring to a Windows network path, such as 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 syntax is in the form:

```\server\sharedfolder\filepath```

Where:
- `server` references the server name in the network (defined by the DNS)
- `sharedfolder` references a label defined by the administrator (for example, admin$ is generally the root directory of the operating system installation).
- `filepath` refers to the subdirectories below the share.
6.12 Managing Triggers

When creating a job, you must specify the conditions (or criteria) that will trigger the job, 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 you can also enable or disable any of the defined triggers. Whenever any of the enabled triggers fire, 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 execute.

You can create the following types of triggers:

- Timer trigger
- File system trigger
- HTTP trigger

A timer trigger allows you to schedule jobs to start at a specific time and run for a specific time interval, with flexible recurring options (such as daily, weekly, on specific days of the week or month, and others).

A file system trigger allows you to monitor a file or directory for changes such as newly added files or modified files (note that deleted files cannot be monitored). You can flexibly configure the directory polling interval (for example, every 60 seconds), and you can optionally set the start and expiry date of the trigger. You can also use wildcards to filter specific files of the directory.

An HTTP trigger allows 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 (for example, every 60 seconds), and you can optionally set the start and expiry date of the trigger.

To add a trigger, click the button corresponding to the trigger type.

Use the following buttons to the right of a trigger to manage it.

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete trigger.</td>
<td>![Delete trigger button]</td>
</tr>
<tr>
<td>Duplicate trigger.</td>
<td>![Duplicate trigger button]</td>
</tr>
<tr>
<td>Undo a previous delete action.</td>
<td>![Undo button]</td>
</tr>
</tbody>
</table>

Some trigger fields have the + and - buttons displayed next to them (for example, the start date of a timer trigger). You can use these buttons to set or clear the value of the trigger field. The value is considered set when it is visible in the page. For example, in the image below, the value of Repeat is not set, while the value of Start is set to "2016-03-29 00:00:00":

![Image showing trigger fields with values set and unset]
Note that, for the set trigger values to take effect, the job must be saved as well.

See also
- Timer Triggers
- File System Triggers
- HTTP Triggers

6.12.1 Timer Triggers

A timer trigger allows you to schedule jobs to start at a specific time and run for a specific time interval, with flexible recurring options (such as daily, weekly, on specific days of the week or month, and others).

The following screen shot illustrates a sample timer trigger.

Timer triggers have the following structure.

<table>
<thead>
<tr>
<th>Run</th>
<th>Defines whether the trigger should fire once or periodically every N number of days. The available options are as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Once</td>
</tr>
<tr>
<td></td>
<td>• Daily</td>
</tr>
</tbody>
</table>
### 6.12.2 File System Triggers

A file system *trigger* allows you to monitor a file or directory for changes such as newly added files or modified files (note that deleted files cannot be monitored). You can flexibly configure the directory polling interval (for example, every 60 seconds), and you can optionally set the start and expiry date of the trigger. You can also use wildcards to filter specific files of the directory.

The following screen shot illustrates a sample file system trigger.

![Sample file system trigger](image-url)

File system triggers have the following structure.

<table>
<thead>
<tr>
<th>Check</th>
<th>Specifies how the trigger should poll the directory or file. Valid options:</th>
</tr>
</thead>
</table>

- On days of week
- On days of months
- On days of 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 field).

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.

Clicking in the Date field opens a pop-up calendar from which you can select the start date.

**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 (see Setting the Default Time Zone).

**Enabled**

The *enabled* check box allows you to enable or disable the trigger. This option is useful when creating and testing new jobs.
The trigger checks the last modification timestamp of all the specified files. If any dates have changed, or a new file has been added, the trigger fires.

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, then the trigger also fires.

### Polling interval
Specifies the frequency, in seconds, with which the directory will be polled.

### Wait N seconds to 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. Clicking in the Date field opens a pop-up calendar from which you can select the start date.

### 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 (see Setting the Default Time Zone).

### Enabled
The enabled check box allows you to enable or disable the trigger. This option is useful when creating and testing 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. When the job runs, FlowForce Server set this parameter to:

a) the file that triggered the job (in case of file system triggers)
b) the name of the temporary file that contains the downloaded content of the polled URI (in case of HTTP triggers).

![Job input parameters](image)

You can pass the value of the triggerfile parameter as input value in any subsequent steps of the job. This way, you can user 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.

For an example of a job that uses the triggerfile parameter, see Creating a Directory Polling Job.

6.12.3 HTTP Triggers

An HTTP trigger allows 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 (for example, every 60 seconds), and you can optionally set the start and expiry date of the trigger.

The following screen shot illustrates a sample HTTP trigger.

![Sample HTTP trigger](image)

HTTP triggers have the following structure.

<table>
<thead>
<tr>
<th>Check</th>
<th>Specifies how the trigger should poll the URI. Valid options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Header date</td>
<td>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).</td>
</tr>
<tr>
<td>Content</td>
<td>Instructs the system to check the Content-MD5 optional HTTP header. This is a 128-bit &quot;digest&quot; used as a message integrity check. If the MD5 header has changed after the polling interval has passed, then the trigger fires. If the header is not provided by the server, the content is retrieved and hashed locally.</td>
</tr>
<tr>
<td>Polling interval</td>
<td>Specifies the frequency, in seconds, with which the URI will be polled.</td>
</tr>
<tr>
<td>Wait N seconds for settle</td>
<td>Defines the time in seconds that the server will wait before starting the next job.</td>
</tr>
<tr>
<td>Start</td>
<td>Defines the trigger's starting date and time. This is an optional field.</td>
</tr>
<tr>
<td>Clicking in the Date field opens a pop-up calendar from which you can select the start date.</td>
<td></td>
</tr>
<tr>
<td>Expires</td>
<td>Defines the date and time when the trigger expires.</td>
</tr>
<tr>
<td>Time zone</td>
<td>Defines the time zone applicable to the start and expiry date and time. The default time zone is defined in the server administration settings (see Setting the Default Time Zone).</td>
</tr>
</tbody>
</table>
### Configuring Jobs

#### Managing Triggers

| Enabled | The *enabled* check box allows you to enable or disable the trigger. This option is useful when creating and testing 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. When the job runs, FlowForce Server sets this parameter to:

a) the file that triggered the job (in case of file system triggers)

b) the name of the temporary file that contains the downloaded content of the polled URI (in case of HTTP triggers).

![Job input parameters](image)

You can pass the value of the **triggerfile** parameter as 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.

For an example of a job that uses the **triggerfile** parameter, see [Creating a Directory Polling Job](#).
6.13 Exposing 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. Click to select the Make this job available via HTTP... check box.
3. Type the name of the Web service in the myURL text box.

At the request URL specified, FlowForce Server starts a listener service which accepts HTTP GET and POST requests. Jobs exposed as services remain active as long as FlowForce server is running.

When the 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 (see Viewing the Job Log).

For a step-by-step example of how to configure such a job, see Exposing a Job as a Web Service.

To view all currently running FlowForce Server Web services:

Open your browser and enter the following URL in the address bar.

```
http://[FlowForceServer][ServerPort]/service/*
```

*[FlowForceServer] and [ServerPort]* refer to the network address and port where FlowForce Server is running. By default, FlowForce Server runs on `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 Setting the Network Address and Port).
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).

If you use Internet Explorer to test FlowForce Server jobs exposed as Web services, you may need to disable the "Show friendly HTTP error messages" option in the Advanced tab.

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 execution behaves like execution via trigger, and is subject to the same queue constraints (see Defining Queue Settings).
6.14 Caching 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>Cache the result</th>
<th>Cache is used whenever this job is called from another job.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a time trigger to create and refresh the cached result or check &quot;Initiated by consumer&quot; option below.</td>
<td></td>
</tr>
<tr>
<td>Create a job that will call this one and will benefit from the cache.</td>
<td></td>
</tr>
<tr>
<td>If &quot;Initiated by consumer&quot; option is chosen then add Refresh Cache timer to prevent cache entries to become too old.</td>
<td></td>
</tr>
<tr>
<td>In case of job input parameters present set &quot;Initiated by consumer&quot; option and set &quot;Maximum number of cache entries&quot; to expected number of possible variations of input parameters.</td>
<td></td>
</tr>
</tbody>
</table>

#### Caching settings

The available caching settings are as follows.

| **Cache the result** | 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. 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). |

---

© 2019 Altova GmbH

### Initiated by consumer

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.

### Maximum number of cache entries

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).

### Auto create a new cache consumer job

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 **Initiated by consumer** option is disabled, the consumer retrieves the actual result returned by the main job. If there is no cached result and the **Initiated by consumer** option is enabled, the consumer retrieves the actual result returned by the main job and also populates the cache.

### Refresh Cache timer

A Refresh Cache timer controls how often the system should refresh the cache of the current job. All currently cached parameter combinations are refreshed.

![Refresh Cache timer](image)

### Purge Cache timer

A Purge Cache timer controls how often the system should purge the cache of the current job.
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)
6.15 Defining Queue Settings

Queue settings enable you to control usage of server resources, for example, by limiting the number of instances running in parallel for each job. In order to change queue settings, your user account must have the "Define execution queues" privilege, see How Privileges Work.

Queue settings

<table>
<thead>
<tr>
<th>Run job using queue:</th>
<th>Select existing queue:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>Define local queue:</td>
</tr>
</tbody>
</table>

Minimum time between runs:

An execution queue provides execution slots, where the number of available slots is governed by the "maximum parallel instances" setting multiplied by the number of workers assigned according to the currently active rule. 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 times the same job may be executed in parallel on the server.

Parallel processing depends on the number of server cores available to FlowForce Server. For example, if you set this value to 12 and only two server cores is licensed, 12 jobs will appear as running in parallel; however, only two jobs will make progress at any given time.

If you are testing parallel service execution from your browser, be aware that the browser may hold back additional requests that address the same URL. As a workaround, you can initiate parallel instances of the same job from two different browsers.
7 Importing and Exporting 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 2019 at 10:51:33 server time would be named `export_20190806105133.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.

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)
- OAuth 2.0 client secret, authorization token and refresh token
- Private keys in a certificate+private key pair

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. Namely, in case of credentials, the password will be empty. In case of certificates, the certificate will not have the private key. In case of OAuth 2.0 credentials, the client secret, the access token, and the refresh token will all be empty.
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, certificate private keys, and OAuth sensitive data 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.

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.

### Importing configuration data

**To import an archive exported previously:**

1. Click **Configuration**, and then click **Import Objects**.
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.

4. Click **Import**.
7.1 Worked Examples

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 examples below are meant to help you understand the implications and how to address them.

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").

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.

**Example 3: Handle missing dependencies**

Let's assume that you have the same two records as above and choose to export only the job, without exporting the credential record:

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>/public/Jobs/AddNumbers</td>
</tr>
<tr>
<td>/public/Jobs/my.credentials</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.
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.
8 Monitoring Job Execution

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. You can view the status of all currently running jobs in the Home page of the FlowForce Web administration interface, in the “Running jobs” section.

<table>
<thead>
<tr>
<th>Instance ID</th>
<th>Job</th>
<th>Activation Time</th>
<th>Last Action</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>public/MyTask</td>
<td>2019-02-05 12:26:00</td>
<td>2019-02-05 12:26:29</td>
<td>Finished successfully after step 2</td>
</tr>
<tr>
<td>10</td>
<td>public/MyTask</td>
<td>2019-02-05 12:27:00</td>
<td>2019-02-05 12:27:00</td>
<td>Running step 2</td>
</tr>
</tbody>
</table>

The Status column reflects the job status as it was when the page was last refreshed. To get the latest status of all jobs, click the Reload Grid button. For reference to all the job instance statuses, see Job Statuses.

In addition to running job instances, the table displays also any recently finished jobs, including jobs that failed for whatever reason. Such jobs are displayed only for a short time (approximately 1-2 minutes) after their execution finished. You can always check the full history of each job instance from the Log page, see Viewing the Job Log.

When it starts, each job instance is assigned a unique ID that is displayed in the Instance ID column. The instance ID helps you subsequently track the execution status of each job instance, from the Log page. You can also click the instance ID inside the table as the job is running—this redirects you to the Log page and displays only the details pertaining to the selected job instance. If you would like to use to the job's unique instance ID in a job (for example, to create unique file names), this is possible with the help of the instance-id expression function.

If necessary, it is possible to stop running job instances by clicking the Stop job button. Note, however, that this may cause data corruption and should be done only exceptionally, see also Stopping Jobs.

If multiple FlowForce Server instances run as a cluster, the "Running jobs" table includes additional details about the cluster members running each job instance, see Monitoring Execution of Distributed Jobs.
8.1 Monitoring Execution of Distributed Jobs

If multiple FlowForce Server instances were configured to run as a cluster, the master FlowForce Server instance is responsible for executing jobs and logging their details. A worker machine does not execute any local jobs and does not have a Log View, unless you convert it back to standalone mode, see also Terminating the "Worker" Mode.

All the logging in a clustered setup is done by the master FlowForce Server instance. In addition, the “Running jobs” table in the Home page includes a Worker column which indicates the cluster member on which the job instance is running. This can be a master or any worker machine that is part of the cluster, depending on the job configuration, see Setting up Distributed Execution.

To view the currently running or recently finished job instances for a specific cluster member (worker or master), click the link in the Worker column. This opens the Cluster member page where you can monitor the job instances running on that cluster member.

Cluster member A0A806AE76E5BA489D4AA86A6E816DC8

Worker

Current Status

Status: Online
Connected since: 2019-02-05 16:20:00
ID: A0A806AE76E5BA489D4AA86A6E816DC8
IP Address: [ ]
Token: [ ] Show Token

Cluster Member Running Jobs

<table>
<thead>
<tr>
<th>Instance ID</th>
<th>Job</th>
<th>Activation Time</th>
<th>Last Action</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>/public/MyDistributedTask</td>
<td>2019-02-05 16:20:00</td>
<td>2019-02-05 16:21:05</td>
<td>Finished successfully after step 1</td>
</tr>
<tr>
<td>60</td>
<td>/public/MyDistributedTask</td>
<td>2019-02-05 16:22:00</td>
<td>2019-02-05 16:22:00</td>
<td>Running step 2</td>
</tr>
</tbody>
</table>

Stop job
8.2 Job Statuses

Across its lifetime, a job instance gets various statuses, as indicated by the Status column in the Running jobs table.

The following table lists all the possible statuses of a job.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created</td>
<td>This is the first state the job is in, before any other action takes place. The next possible states are Starting, Waiting, Waiting for slot, Failed, and Aborted.</td>
</tr>
<tr>
<td>Waiting</td>
<td>This status may happen for a job instance triggered as a result of a file system or HTTP trigger. It indicates that the job instance was triggered but</td>
</tr>
<tr>
<td>Status</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>the settle period specified by the Wait N seconds for settle option has not elapsed yet.</td>
<td></td>
</tr>
<tr>
<td>Waiting for slot</td>
<td>Indicates that the job instance was triggered but the execution queue is currently full. An execution queue has a limited number of slots, so only the specified number of job instances can execute in parallel on the same queue, see also Defining Queue Settings. Any further instances arriving for that queue will wait until a slot becomes free, and will appear as having the Waiting for slot status. Usually, the next status after Waiting for slot is Starting, unless interruptions occur.</td>
</tr>
<tr>
<td>Starting</td>
<td>This status is usually very brief; it means that the job instance has been assigned a free queue slot and will soon start executing the first step.</td>
</tr>
<tr>
<td>Running</td>
<td>Indicates that the job instance is currently running and will stay in this state until execution is complete or until some external event occurs that ends execution prematurely. Except for a very brief time window in the beginning, this status has a step number associated with it, such as Running step {step}.</td>
</tr>
<tr>
<td>Failed</td>
<td>Execution of this job instance has finished and was considered a failure. This is a final status and there will be no further attempts to run the job instance.</td>
</tr>
<tr>
<td>Finished successfully</td>
<td>This is a final state which indicates that the job has completed successfully.</td>
</tr>
<tr>
<td>Aborting</td>
<td>The job instance is attempting to finish execution because a FlowForce user has stopped it by clicking the Stop job button. Note that stopping the job is an asynchronous action which may take time, and a job instance may actually be able to complete successfully before it transitions to the Aborted state. If this happens, the job will be reported as having finished successfully. If the previous status had a step number, then the Aborting after step {step} status is shown instead of Aborting.</td>
</tr>
<tr>
<td>Aborted</td>
<td>The job instance has finished execution. Normally, this status indicates that a user has stopped the job, although it may also occur indirectly after an unexpected shutdown. This is a final state and it indicates that at least some part of the job hasn't finished. If the previous status had a step number, then the Aborted after step {step} status is shown instead of Aborted.</td>
</tr>
<tr>
<td>Interrupted</td>
<td>The execution of the job instance was interrupted (for example, because a network cable was unplugged, the worker machine has crashed or was shut down, and similar force-terminated events). This is a more forceful variation of the Aborted state. The job instance cannot be restarted so it should be</td>
</tr>
</tbody>
</table>
## Statuses

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>treated as failed. There is no guarantee about the consistency of the job, so it is recommended that you check its outcome manually.</td>
</tr>
<tr>
<td>Lost connection</td>
<td>Applicable when multiple FlowForce instances run as a cluster. This status indicates that the master machine has lost the connection to the worker machine. There is currently no way to reasonably determine the status of this job instance. Better state information may become available when the worker is back.</td>
</tr>
<tr>
<td>Synchronizing</td>
<td>Applicable when multiple FlowForce instances run as a cluster. In a clustered setup, the master machine periodically gets the current progress of job instances from the worker machines. This status indicates that the connection to the worker has been restored after it was lost, and FlowForce is attempting to obtain the latest status from the worker.</td>
</tr>
<tr>
<td>Untracked</td>
<td>Indicates that the FlowForce Server process is currently unable to track the job. This status may occur when the master unexpectedly went down while the job was still running on a worker machine. There is a subtle difference between <strong>Interrupted</strong> and <strong>Untracked</strong>: In the former case, FlowForce Server knows it was terminated as a process, whereas in the latter case it just isn't aware of what exactly has happened. Like with <strong>Interrupted</strong>, there is no guarantee about the consistency of the job, so it is recommended that you check its outcome manually.</td>
</tr>
<tr>
<td>Superseded</td>
<td>This status signifies an uncritical termination of a job before the job had actually the chance to do something. This is not a critical condition. It essentially means that FlowForce Server has detected a further change and thus canceled the previous job instance and created a new one. For example, this status may happen for a job instance triggered as a result of a file system or HTTP trigger. If the file triggering the job has changed before the settle period specified by the <strong>Wait N seconds for settle</strong> option, then the job instance may get this status. This status can also occur with time triggers, if the time trigger was triggered again while a previously started instance is still waiting for a slot.</td>
</tr>
</tbody>
</table>
8.3 Stopping Jobs

You can stop any currently running job provided that you (or any roles you are member of) have the *Stop any job* privilege (see *How Privileges Work*).

**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.

![Running Jobs Table]

2. Click **Stop job**. FlowForce Server prompts you to confirm the action:

![Stop Running Instance Confirmation]

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”.

---

© 2019 Altova GmbH

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.
8.4 Viewing the Job Log

You can quickly find out the details logged by FlowForce Server with respect to a particular job from the Log View. You can access the Log View 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).

Log View

As illustrated above, the Log View provides sorting, pagination, and filtering of records. By default, it displays the logged events of the last 7 days, with minimum severity set to "Info". To filter out information messages from the log, select "Warning" as minimum severity. To skip both information messages and warnings, select "Errors" as minimum severity.

A unique job instance exists for each run of job, as shown by the Instance column. You can filter out log entries either by the unique instance ID of the job, or by the job (object) path. In the latter case, the log will display all instances pertaining to the selected job path.

After changing any of the filters, click Show to update the page with your selection. To refresh the log, click the Reload Grid button.

Note: 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 job configuration page or to a details page that displays further information about the logged event. Be aware that there is a default threshold that specifies the maximum length of a log entry. If you find out that certain log entries (such as parameter values in steps) are truncated because they are too long, this can be addressed, see Changing the Logging Settings.
9 The FlowForce Expression Language

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 a 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.
9.1 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).
9.2 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/file/system/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.

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:

Therefore, an expression such as 1+1 is not a valid in a string field, because it is implicitly evaluated as numeric. On the other hand, the expression ‘1+1’ is valid in a string field. Consider the following examples:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Will be evaluated as...</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</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>
<tr>
<td>Expression</td>
<td>Will be evaluated as...</td>
<td>Explanation</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<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>
<tr>
<td><code>{content(stdout(result))}</code></td>
<td>[...] (as String)</td>
<td>This expression uses two nested expression functions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The function <code>stdout</code> gets the standard output of a shell command, as stream.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The function <code>content</code> converts the stream value to a string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Although the expression is correct, the job will validate successfully only when the following is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The value &quot;result&quot; has been previously declared.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The value &quot;result&quot; actually contains the standard output of a shell command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The expression is embedded into a string field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See also <a href="#">Calling Expression Functions</a>.</td>
</tr>
</tbody>
</table>
9.3 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:

![Example of an execute function with string field](image1.png)

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>
9.4 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 (that is, from within an expression).

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.
9.5 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(output)}` 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.
9.6 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 \text{true} (&lt;br/&gt;2 + 3 == 4) computes to \text{false}</td>
</tr>
</tbody>
</table>
| `!=`     | Checks if \(a\) and \(b\) are not equal. Note that the following three expressions are equivalent:  
          - \(a != b\)  
          - \(\text{not} (a == b)\)  
          - \(a <> b\) | \(2 + 2 != 5\) computes to \text{true} \(<br/>3 + 2 != 5\) computes to \text{false} |
| `<`      | Checks if \(a\) is less than \(b\) (numerically less for numbers, see below for strings). | \(4 < 5\) computes to \text{true} |
| `<=`     | Checks if \(a\) is less than or equal to \(b\). | \(5 <= 5\) computes to \text{true} |
| `>`      | Checks if \(a\) is greater than \(b\). | \(5 > 1\) computes to \text{true} |
| `>=`     | Checks if \(a\) is greater than or equal to \(b\). | \(5 >= 5\) computes to \text{true} |
| `+`      | Addition. | \(1 + 1\) computes to 2 |
| `-`      | Subtraction. | \(2 - 1\) computes to 1 |
| `*`      | Multiplication. | \(3 * 2\) computes to 6 |
| `/`      | Division. | \(6 / 3\) computes to 2 |

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.
10 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, AS2 integration, 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.transformed</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.
10.1 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.
- 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 on the "Options" dialog box in MapForce, for example:

![Options dialog box](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.
Preparing Files for Server Execution

To make paths portable to non-Windows operating systems, use relative instead of absolute paths when designing the mapping in MapForce. For example, you can copy all input or output files required by the mapping into the same directory as the mapping, and then reference them just by file name. 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.
10.2 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 the MapForce and StyleVision documentation (https://www.altova.com/documentation).

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:

   ![Image of the FlowForce Server interface showing the creation of a job from a mapping or transformation]

   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.

   ![Configuration dialog](image)

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:

   ![Execution steps](image)

   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.
10.2.1 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 following fields are considered sensitive data:

- **Password** (for credentials of type "Password")
- **Client Secret, Access Token, and Refresh Token** (for credentials of type "OAuth 2.0")

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:

```
Name: airportCode  Type: string  optional
Name: AirportStatus (output)  Type: string  Default: AirportStatus.json
Name: my.credentials  Type: credential
Name: Working-directory  Type: string as directory  Default: 
```

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.

In case of mapping functions that require OAuth 2.0 authorization, the access token may expire or be revoked by the Web service provider at any time. When this happens, FlowForce Server attempts to acquire a new one automatically while the job instance runs. If multiple running jobs use the same credential and if the runtime factors allow it, FlowForce Server will refresh the access token in a centralized manner and synchronize all the affected job instances accordingly.

10.2.2 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

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.

### 10.2.3 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 ([https://www.altova.com/documentation](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.

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:** Overiding 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.
### 10.3 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 `SimpleTotal.mapping` is being executed.

Notice that the job consists of two steps:

1. Step 1 calls MapForce Server to actually run the `SimpleTotal.mapping` function. Importantly, the **Assign this step’s result to** field gives a name to the mapping result (in this case, it is `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 `result`. In order for the output to become useful, `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 `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, `output` is the name you entered in the **Assign this step’s result to** to field.

<table>
<thead>
<tr>
<th>FlowForce Expression</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>stdout(output)</code></td>
<td>Converts <code>output</code> to a stream.</td>
</tr>
<tr>
<td><code>content(stdout(output))</code></td>
<td>Converts <code>output</code> to string.</td>
</tr>
</tbody>
</table>
Integration with Altova Product Line

Accessing the Mapping/Transformation Result

<table>
<thead>
<tr>
<th>FlowForce Expression</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>as-file(stdout(output))</td>
<td>Converts output to a file.</td>
</tr>
<tr>
<td>as-file(nth(results(output), 0))</td>
<td>This kind of expression is required if output 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 output to a sequence of streams, picks up the first stream from the sequence, and converts it to file. For an example, see Creating a Job from a StyleVision Transformation.</td>
</tr>
<tr>
<td>as-file(nth(results(output, &quot;CompletePO&quot;), 0))</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. For an introduction to FlowForce expressions, see The FlowForce Expression Language.
10.4 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/2020/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.
11 AS2 Integration

AS2 (Applicability Statement 2) is a specification that enables exchanging files securely over the Internet. AS2 is used by businesses to exchange primarily EDIINT (EDI over Internet) and XML files through either HTTP or HTTPS.

This documentation includes references to the following publications:

- RFC 4130, "MIME-Based Secure Peer-to-Peer Business Data Interchange Using HTTP, Applicability Statement 2 (AS2)", see https://www.ietf.org/rfc/rfc4130.txt

Main Features

- With FlowForce Server Advanced Edition, you can send messages in AS2 format to your organization's AS2 trading partners by means of FlowForce jobs. You can also receive AS2 messages from trading partners and further process or store them as required, effectively turning FlowForce Server into an AS2 Server.
- You can optionally encrypt and sign AS2 messages sent to partners, with the help of digital certificates. To support encryption and signing (both as an AS2 data sending or receiving partner), FlowForce Server has a certificate store where you can import and manage centrally the public certificates received from all trading partners, and the public+private certificate pairs created by your organization. As a result, when you receive from other trading partners signed and encrypted AS2 messages, FlowForce Server can decrypt and verify the signature of such messages. Likewise, when you send encrypted and signed data, FlowForce Server prepares this data using the respective certificates previously imported into its store.
- From FlowForce, you can optionally request that the partner send a synchronous Message Disposition Notification (MDN) in reply to an AS2 message sent from FlowForce Server. You can also request that the partner sign the MDN. When FlowForce Server acts as receiver of AS2 messages, it sends MDNs automatically in reply to received AS2 requests.
- FlowForce Server can encrypt and decrypt data using any of the following algorithms: DES, 3DES, AES-128, AES-192, AES-256, RC2-40, RC2-64, RC2-128, RC4-40, RC4-128. It can sign or verify signed data using any of the following algorithms: MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512.
- Optionally, you can enable compression of sent messages (and you can flexibly specify if compression should occur before or after signing). When you receive compressed AS2 data from other trading partners, FlowForce Server automatically performs decompression of data if necessary (regardless of whether data was compressed before or after signing).
- You can integrate jobs that send or receive AS2 data into your business data flows and customize them just like any other FlowForce jobs. For example, jobs can be triggered on demand or in a scheduled manner, have multiple execution steps, conditional processing, user access rights, and so on. In addition to this, they benefit from all the functionality provided by FlowForce Built-in Functions and FlowForce Expression Functions.

Limitations

- Currently, FlowForce supports only synchronous MDNs (Message Disposition Notifications). Asynchronous MDNs are not supported.
- The size of messages is limited by available system memory.
- Basic HTTP authentication is supported (preemptive, credentials are included in the initial request). Digest authentication, or HTTPS authentication by means of client certificates are not supported.
- Import of PEM files that contain only the private key (without certificate) is not supported.
11.1 Overview: Sending AS2 Data

The diagram below illustrates the high-level process of sending AS2 messages with FlowForce Server Advanced Edition.

Sending AS2 data with FlowForce Server

The process illustrated above works as follows:
<table>
<thead>
<tr>
<th>Step #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Configure AS2 partner and certificates</td>
</tr>
<tr>
<td>2.</td>
<td>Create a job</td>
</tr>
<tr>
<td>3.</td>
<td>Job runs and sends AS2 message</td>
</tr>
<tr>
<td>4.</td>
<td>Partner replies with synchronous MDN</td>
</tr>
</tbody>
</table>

To set up the communication with AS2 partners, you will need to obtain their AS2 connectivity details (such as URI and AS2 name), and exchange certificates. The certificates must be imported (and partner details must be entered) into FlowForce Server, see Configuring AS2 Certificates and Configuring AS2 Partners.

A FlowForce job must be created in order to send the AS2 message. The FlowForce Server job may be configured to run in various ways, depending on your business needs. For example, it can run as a Web service call, or whenever a file changes on the file system, or it could be scheduled to occur at a specific time and date, see also Managing Triggers.

In order to send the AS2 message, your job (or execution step within a job) must call the FlowForce Server built-in function /system/as2/send. This function takes a number of parameters required to send the AS2 message, including the partner object configured in step 1 and the AS2 message content that you want to send. Your job may also need to call various FlowForce Server expression functions in order to convert the mapping output to the required form (for example, from a file to a stream), see Stream Functions.

When you create the AS2 partner object in step 1, you may optionally request that a Message Disposition Notification (MDN) be sent by the partner in reply to the AS2 message sent by FlowForce Server. The partner must send the MDN in the same session as the HTTP call outgoing from FlowForce Server (that is, it must be configured as "synchronous").

The diagram above represents a simple configuration. It assumes that the content required for the AS2 message is readily available and must only be supplied as input to the FlowForce Server job. If you need to generate the AS2 message content automatically by mapping data from various sources, the AS2 process can be further automated with Altova MapForce and MapForce Server, see AS2 Integration with MapForce and MapForce Server.

For step-by-step instructions, see Sending AS2 Messages.
11.2 **Overview: Receiving AS2 Data**

The diagram below illustrates the high-level process of receiving messages with FlowForce Server Advanced Edition.

![Diagram of receiving AS2 data](image)

Receiving AS2 data with FlowForce Server

The process illustrated above works as follows:
### AS2 Integration Overview: Receiving AS2 Data

<table>
<thead>
<tr>
<th>Step #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Configure AS2 partner and certificates</td>
<td>To set up the communication with AS2 partners, you will need to obtain their AS2 connectivity details (such as URI and AS2 name), and exchange certificates. The certificates must be imported (and partner details must be entered) into FlowForce Server, see Configuring AS2 Certificates and Configuring AS2 Partners.</td>
</tr>
<tr>
<td>2. Create a job</td>
<td>A FlowForce job must be created in order to expose the AS2 service where FlowForce will listen for AS2 requests.</td>
</tr>
<tr>
<td>3. Partner sends AS2 data</td>
<td>Once you've shared the URL of the service with your partners, they can start sending AS2 requests to it.</td>
</tr>
<tr>
<td>4. Process incoming AS2 data</td>
<td>Upon receiving the AS2 message, FlowForce attempts to decrypt and validate it. If this fails, FlowForce sends an error MDN before starting the job. Otherwise, the incoming data is processed by the job that exposes the AS2 service. You can configure the job to process data according to your needs (for example, convert the message from stream to string, read specific headers from the message, save data to a file with a custom name, get the name of the sending partner, and so on). According to AS2 specification, the MDN should concern only the delivery of the message, not the content of the message. For this reason, the AS2 receiving job must be as minimal as possible (typically, saving the message to a file or a database). The AS2 receiving job should never fail because of reasons related to the content of the message. Therefore, any extra steps (other than accepting the message and saving it) must be defined as separate jobs. Otherwise, if the receiving job contains a step not related to message delivery and that step fails, this will lead to a failure (negative) MDN in turn, which is not expected to happen according to the AS2 specification.</td>
</tr>
<tr>
<td>5. Reply with synchronous MDN</td>
<td>After FlowForce Server finished processing the job, it sends back a synchronous MDN to report either success or failure based on job execution result.</td>
</tr>
</tbody>
</table>

For more information about configuring FlowForce as an AS2 server, see Receiving AS2 Messages.
11.3 AS2 Integration with MapForce and MapForce Server

FlowForce Server Advanced Edition provides the functionality required to send AS2 messages to trading partners, or receive AS2 from trading partners. In addition, FlowForce Server is capable of processing AS2 data and storing it locally, with the help of its built-in set of functions. For even more advanced needs, if you need to prepare AS2 data from some existing source (for example, a database), or convert it to other formats, or send it to some Web service, you can also include MapForce and MapForce Server into the AS2 process.

There are two main scenarios where MapForce and MapForce Server are necessary:

1. To map or generate data in any format supported by MapForce (such as XML, XBRL, Excel, databases, Web services), before sending it to AS2 partners.
2. To transform data received from AS2 partners in a variety of ways (for example, convert it to Excel, convert it to a different XML schema, store it in a database, send it to a Web service, and so on).

Generating and sending AS2 data

In a scenario where you need to prepare or generate AS2 data with MapForce before sending it to partners, the high-level process looks as follows:
Generating and sending AS data

In the diagram above, both MapForce Server and FlowForce Server must be installed on the same machine (it can be a Windows, Linux, or macOS operating system, see System Requirements). MapForce may run on the same machine as MapForce Server and FlowForce Server (provided that it’s a Windows machine), or on a different machine that can connect to FlowForce Server via HTTP or HTTPS. The AS2 partner is a remote server with which FlowForce Server communicates through HTTP(S).

The AS2 process illustrated above works as follows:
<table>
<thead>
<tr>
<th>Step #</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1. | Design and test the EDI/XML data mapping  
With MapForce, you can design a data mapping transformation that takes as input data in various formats (including plain text, CSV, JSON, XML, various EDI flavors, databases, Web services) and outputs one or several files in a destination format (for example, UN/EDIFACT). Designing a mapping for EDI purposes is not different to other mappings, and various such examples are included in MapForce documentation, see the EDI chapter. While you design the mapping, you can validate and preview the mapping output directly in MapForce, by clicking the Output tab. To ensure that the mapping is suitable for execution in a server environment, you will need to design and test it for the BUILT-IN transformation language. |
| 2. | Deploy mapping to FlowForce Server  
FlowForce Server automates various tasks by means of on demand or scheduled jobs that can be defined from a Web interface. FlowForce Server can also automate the execution of a mapping designed with MapForce, provided that MapForce Server runs under FlowForce Server management. Once the MapForce mapping produces the required output, you are ready to automate its execution, by deploying it to FlowForce Server. |
| 3. | Configure AS2 partner and certificates  
To set up the communication with AS2 partners, you will need to obtain their AS2 connectivity details (such as URI and AS2 name), and exchange certificates. The certificates must be imported (and partner details must be entered) into FlowForce Server, see Configuring AS2 Certificates and Configuring AS2 Partners. |
| 4. | Create a job  
A FlowForce job must be created in order to (a) run the mapping and produce the required output, and (b) send the AS2 message (see also step 7). These two actions may be either execution steps of the same job, or two different jobs altogether. For an example of a FlowForce Server job that runs a MapForce mapping, see Creating a Job from a MapForce Mapping. |
| 5. | Run job  
The FlowForce Server job created in the previous step may be configured to run in various ways, depending on your business needs. For example, it can run as a Web service call, or whenever a file changes on the file system, or it could be scheduled to occur at a specific time and date, see also Managing Triggers. This step is fully automated. |
| 6. | Run data mapping  
This step also takes place automatically and is executed by MapForce Server. If a job is configured to execute a data mapping (be it scheduled or on demand), an internal call to MapForce Server takes place. As a result, MapForce Server runs the mapping and returns the output to FlowForce Server. |
| 7. | Pick output and send AS2 message  
In order to send the AS2 message, your job (or execution step within a job) must call the FlowForce Server built-in function /system/as2/send. This function takes a number of parameters required to send the AS2 message, including the partner |
<table>
<thead>
<tr>
<th>Step #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>object configured in step 3, the partner’s URI, and the AS2 message content that you want to send. Your job may also need to call various FlowForce Server AS2 expression functions in order to convert the mapping output to the required form (for example, from a file to a stream).</td>
</tr>
<tr>
<td>8. Partner replies with synchronous MDN</td>
<td>When you create the AS2 partner object in step 3, you may optionally request that the partner send a Message Disposition Notification (MDN) in reply to the AS2 message sent by FlowForce Server, see also AS2 Concepts. The partner must send the MDN in the same session as the HTTP call outgoing from FlowForce Server (that is, it must be configured as “synchronous”).</td>
</tr>
</tbody>
</table>

Receiving and processing AS2 data

If your organization receives AS2 data from trading partners, you can additionally configure a data receiving workflow. In this scenario, your organization would be able to not only receive and store AS2 data, but also transform it to other formats, save it to a database, or send it to another Web service. For example, you could receive files in EDI or XML format from AS2 trading partners and then supply them as input to some mapping that runs as a recurrent FlowForce Server job. In this scenario, an example AS2 process looks as follows:
Receiving and processing AS2 data

The example AS2 process illustrated above works as follows:

<table>
<thead>
<tr>
<th>Step #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3</td>
<td>These are the same steps as in the previous table. The only difference is that this time the mapping is expected to take as input some file that your organization expects to receive from an AS2 trading partner (for example, an EDI or XML file).</td>
</tr>
<tr>
<td>Step #</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>4. Create a job (AS2 service)</td>
<td>This is a one-time step. In this step, you create a FlowForce Server job that exposes an AS2 service. The AS2 service listens for requests from your AS2 partners at a configured HTTP(S) address and port.</td>
</tr>
<tr>
<td>5. Send AS2 data</td>
<td>In this step, a trading partner submits AS2 messages to the AS2 service. For communication to be successful, the partner’s AS2 name and certificates must already be defined in FlowForce Server.</td>
</tr>
<tr>
<td>6. Reply with synchronous MDN</td>
<td>FlowForce Server replies to the AS2 partner with a synchronous MDN that indicates the outcome of the operation (success or error).</td>
</tr>
<tr>
<td>7. Process and save data</td>
<td>As soon as there is an incoming message, a FlowForce Server job converts the received data to a string or a file, and then stores it in some directory, or passes it to another job as argument. The exact processing logic is configurable with the help of FlowForce Server built-in and expression functions.</td>
</tr>
<tr>
<td>8. Run data mapping</td>
<td>The FlowForce Server job that receives AS2 data may optionally invoke the data mapping job that was created in the first step. The mapping job takes as input the AS2 data received from the partner and then processes it in any of the ways supported by MapForce: for example, transforms it to another format, saves it to a database, sends it to another Web service, and so on.</td>
</tr>
</tbody>
</table>
11.4 AS2 Concepts

In order to send AS2 messages to a trading partner, you must first obtain from the trading partner the AS2 connectivity details, including any digital certificates required for data encryption and signing. Also, the following must be established:

- Does the partner require connections over HTTP or HTTPS?
- Does the partner require that AS2 messages be encrypted?
- Does the partner require that AS2 messages be signed?
- Do you need a confirmation (MDN, from "Message Disposition Notification") from the partner that the AS2 message has been received?

HTTP(S) connection

The HTTP connection encryption is different from (and should not be confused with) the encryption of the actual AS2 message. Your trading partner might accept plain HTTP and not require HTTPS connections at all, because the AS2 message is typically already encrypted separately on a different layer (see the next paragraph). If the trading partner requires that AS2 messages be sent over HTTPS instead of plain HTTP, then the server of your trading partner is most likely already configured to accept SSL-encrypted connections from clients, and no additional configuration should be necessary on your side.

AS2 encryption

"Encryption" of the AS2 message means changing (enciphering) data before transmitting it, in such a way so that only the intended party (that is, your trading partner) can decipher it and read it. Note that the AS2 message encryption certificates are not the same as the certificates used to secure the connection to the trading partner (see previous paragraph). To make AS2 message encryption possible, you must have the trading partner's public certificate and add it to the FlowForce Server certificate store, see Configuring AS2 Certificates.

AS2 signing

"Signing" means adding to the message a digital signature, which only the signer of the message (that is, your organization) could have created for this particular message, but which everyone (in particular, your trading partner) can verify – provided they know your organization’s public certificate. Therefore, you must add your organization’s private certificate (or private key) to the FlowForce certificate store, see Configuring AS2 Certificates, and send your public signature verification certificate to your trading partner.

MDN

Message Disposition Notifications (MDNs) act as receipts in AS2 communication. By requesting a signed notification, you can verify that your message was received untampered and accepted for processing. AS2 supports both synchronous MDNs (as response to the HTTP request) and asynchronous MDNs (delivered by a separate mechanism, not necessarily HTTP). FlowForce Server will always request a synchronous MDN, optionally signed, see Configuring AS2 Partners. Requesting asynchronous MDNs is currently not supported, see the Limitations.

Once you have agreed with the trading partner how data is to be sent and exchanged the required certificates, the next step is to add the relevant certificates and partner details to FlowForce Server (see Configuring AS2 Certificates and Configuring AS2 Partners, respectively).
11.5 Configuring AS2 Certificates

Digital certificates provide security at various levels in the AS2 message exchange process. In the context of AS2 communications, certificates may be used for (but are not limited to) the following purposes:

- AS2 message encryption
- AS2 message signing
- AS2 signature verification

FlowForce Server has a certificate store that is independent from the certificate store of the operating system where FlowForce Server runs. In FlowForce Server, certificates are stored in containers (and thus benefit from the same user access mechanism as other objects across FlowForce, see How Permissions Work). All the private or public certificates that you need for AS2 process must be imported into FlowForce Server (you can decide what the target containers should be and which users should be able to access them).

For AS2 message encryption and signature verification, the configuration steps are as follows:

1. Obtain from your trading partner the public certificate used for encryption or signature verification. This will often be the same certificate.
2. Import the certificate into the FlowForce Server certificate store, as shown below. You will need to refer to this certificate when creating the partner details in FlowForce (see Configuring AS2 Partners).

For AS2 message decryption and signing, the configuration steps are as follows:

1. Create your organization's public certificate, and the private key (in a program external to FlowForce Server). If your organization's certificate for signing already exists in the certificate store of the operating system, then export it to a file (the file must contain both the public certificate and the private key). For instructions on how to do this on Windows, see https://technet.microsoft.com/en-us/library/cc754329(v=ws.11).aspx. For Linux, the certificate files must be copied from the directory which acts as certificate store, for example /etc/ssl/private or /etc/ssl/certs on Ubuntu. For macOS, see https://support.apple.com/kb/PH20122?locale=en_US.
2. Send the public certificate (without the private key) to the partner. The private key must not be shared with anyone outside of your organization.
3. Import the certificate (with the private key) into the FlowForce Server certificate store, as shown below. If the partner will send signed MDNs, then the partner's public certificate (required to verify the MDN signature) must also be imported into FlowForce. Again, you will need to refer to this certificate when creating the partner object, see Configuring AS2 Partners.

To import a certificate into FlowForce Server:

1. Log on to FlowForce Server Web Administration Interface.
2. Click Configuration, and then navigate to the container in which you want to create the certificate.

**Note:** By default, the "Public" container is accessible to all authenticated FlowForce Server users and so it might not be a suitable place to store sensitive information. It is recommended that you either restrict access to the "Public" container, or define sensitive objects in a separate container to which only entitled users have permissions, see Permissions and Containers.

3. Click Create, and then Create Certificate.
4. Enter a name, and, optionally, a description for the certificate. Choose a descriptive name to easily identify the certificate later. The description can be changed later.

5. Click **Browse** and select the certificate file.

The imported file must be in PEM, DER, or PKCS#12 format (this should not be confused with the file extension). The file extension can be one of the following: .pem, .der, .cer, .crt, pfx, p12. FlowForce will treat the file as follows:

- File is treated as PEM format if extension is .pem, .cer, .crt, and the file contains a line that starts with "-----BEGIN " or "---- BEGIN ".
- File is treated as DER format if extension is .der, .cer, .crt and the file does not contain the line above.
- File is treated as PKCS#12 if extension is .p12 or .pfx.

Files that contain only a private key (but not the certificate) cannot be imported.

6. If the certificate file contains a private key that requires a password, enter the password into the corresponding field. If the certificate file contains an unprotected private key, click **Delete** to omit this field.

7. Click **Save**.

If the certificate was successfully imported, its details are displayed in the page, for example:
Since certificates expire after a certain amount of time, you will also need to periodically replace them from the FlowForce Server Web administration interface. This applies both to certificates created by your organization and those you received from your trading partner. (It is assumed that your trading partner will inform you when their public certificate expires, and send you the new certificate. Likewise, you should inform the trading partner when your public certificate expires and send them the new one.) The certificate's expiration date and other
related information can be viewed from the Web administration interface (after you imported the certificate into FlowForce Server).

When you replace a certificate in FlowForce Server, the change will affect any partners using this certificate. To ensure the integrity of your AS2 operations, always co-ordinate changes to your organization's certificates with your trading partners in advance.

To replace a certificate:

1. After logging in to FlowForce Server, click Configuration, and then navigate to the container where the certificate is stored.
2. Click the certificate entry. The certificate details page loads.
3. Click Import certificate.
4. Click Browse and select the new certificate.
5. Click Save. This replaces the old certificate with the new one.

Certificates previously imported into FlowForce Server can be deleted just like other FlowForce Server objects (select the check box next to the specific record, and then click Delete). Cloning or exporting certificates is not possible.

For an example of an AS2 exchange which involves two trading partners that exchange certificates for signing and encryption, see Example: Full AS2 Message Exchange (Advanced).
11.6 Configuring AS2 Partners

The term "Partners" refers to parties taking part in AS2 communications, that is, your organization and your organization's trading partners. In order for your organization to communicate with any AS2 trading partners, their details must first be defined in FlowForce Server. Once you define the AS2 partner details, they can be reused later in jobs. Namely, when you create jobs that send AS2 messages, you will be able to select the partner from a list of trading partners already defined (instead of having to enter the partner details for each FlowForce job).

**Note:** If encryption and signing must be enabled, make sure to import the required certificates (your organization's and your partner's) into FlowForce Server, see [Configuring AS2 Certificates](#).

**To configure the AS2 partner:**

1. Log on to FlowForce Server [Web Administration Interface](#).
2. Click Configuration, and then navigate to the container in which you want to create the partner object.

**Note:** By default, the "Public" container is accessible to all authenticated FlowForce Server users and so it might not be a suitable place to store sensitive information. It is recommended that you either restrict access to the "Public" container, or define sensitive objects in a separate container to which only entitled users have permissions, see [Permissions and Containers](#).

3. Click Create, and then Create AS2 Partner.

The settings in the partner configuration page are organized in groups and have the same behavior as in other parts of the FlowForce Web administration interface. For example, if a group is optional, you must first click ![+] to set the required options. To make the group optional again, click the ![−] button—this hides this group of settings and makes it irrelevant.

The partner configuration page consists of the following groups of settings:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partner Name</td>
<td>Required field. A name that identifies the trading partner to FlowForce Server. This name appears throughout the FlowForce graphical user interface to help you identify this trading partner.</td>
</tr>
<tr>
<td>Partner Description</td>
<td>Optional field. Free description text about the partner organization (for example, postal address, contact person, and so on).</td>
</tr>
</tbody>
</table>

**Partner Settings**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| AS2 name| Required field. When FlowForce Server sends AS2 data, this value identifies the receiver of the data exchange (the value of the "AS2-
### Local Side Settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS2 Name</td>
<td>Required field. When FlowForce Server sends AS2 data, this value identifies</td>
</tr>
<tr>
<td></td>
<td>the sender of the data exchange (the value of the &quot;AS2-From&quot; header).</td>
</tr>
<tr>
<td></td>
<td>When FlowForce Server receives AS2 data, this value identifies the receiver</td>
</tr>
<tr>
<td></td>
<td>of the data exchange (the value of the &quot;AS2-To&quot; header).</td>
</tr>
<tr>
<td></td>
<td>This name is usually agreed between AS2 trading partners and must be unique</td>
</tr>
<tr>
<td></td>
<td>system-wide, see also RFC 4130, §6.2.</td>
</tr>
</tbody>
</table>

### AS2 Service Settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive messages</td>
<td>Optional field. Select this check box to allow FlowForce Server to</td>
</tr>
<tr>
<td></td>
<td>receive messages from this AS2 partner.</td>
</tr>
<tr>
<td></td>
<td>If you are creating an AS2 partner to whom you will only be sending AS2</td>
</tr>
<tr>
<td></td>
<td>data and from whom you will not receive AS2 data, clear this check box.</td>
</tr>
<tr>
<td></td>
<td>This helps avoid errors when there is more than one partner with the same</td>
</tr>
<tr>
<td></td>
<td>&quot;Local AS2 Name&quot; and &quot;AS2 Name&quot; pair. If that happens, you will be able to</td>
</tr>
<tr>
<td></td>
<td>receive AS2 messages only from the partner for which this check box is</td>
</tr>
<tr>
<td></td>
<td>selected.</td>
</tr>
</tbody>
</table>

### HTTP Endpoint Settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request URL</td>
<td>Required field. This field must specify the partner URL to which AS2</td>
</tr>
<tr>
<td></td>
<td>messages will be sent, for example:</td>
</tr>
<tr>
<td></td>
<td><a href="http://example.org:8080/as2/HttpReceiver">http://example.org:8080/as2/HttpReceiver</a>.</td>
</tr>
<tr>
<td></td>
<td>The value must start with &quot;http:&quot; or &quot;https:&quot;.</td>
</tr>
<tr>
<td>Redirect Mode</td>
<td>Optional field. For security reasons, you may want to disallow that</td>
</tr>
<tr>
<td></td>
<td>HTTP requests be redirected, or only allow redirection on the same</td>
</tr>
</tbody>
</table>
### Compression Settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Compression</td>
<td>Optional field. Select this check box if FlowForce Server should compress AS2 data before sending it to partner.</td>
</tr>
</tbody>
</table>

### Security Settings | Encryption

This group of settings must be defined if your organization should encrypt AS2 messages sent to this partner.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Optional field. Specifies the symmetric algorithm to be used for encryption. Valid values:</td>
</tr>
<tr>
<td></td>
<td>- DES</td>
</tr>
<tr>
<td></td>
<td>- 3DES [Default]</td>
</tr>
<tr>
<td></td>
<td>- AES-128</td>
</tr>
<tr>
<td></td>
<td>- AES-192</td>
</tr>
<tr>
<td></td>
<td>- AES-256</td>
</tr>
<tr>
<td></td>
<td>- RC2-40</td>
</tr>
</tbody>
</table>
Configuring AS2 Partners

Security Settings | Decryption
This group of settings must be defined if your organization should decrypt AS2 messages received from this partner.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Partner Certificate** Required field. Specifies the certificate to be used for AS2 message encryption. This must be a public certificate that you received from your trading partner and then imported into FlowForce Server, see Configuring AS2 Certificates.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Algorithm**          Optional field. Specifies the algorithm(s) that a partner is allowed to use when encrypting messages sent to your organization.  
If the trading partner uses another algorithm or one that is not selected, then FlowForce Server will send an error MDN and the job will not be started. The error MDN in this case includes a text like: "automatic-action/MDN-sent-automatically ; failed / error: insufficient-message-security"   
Valid values for this field are: |
|                        | - DES                                                                        |
|                        | - 3DES                                                                       |
|                        | - AES-128                                                                    |
|                        | - AES-192                                                                    |
|                        | - AES-256                                                                    |
|                        | - RC2-40                                                                     |
|                        | - RC2-64                                                                     |
|                        | - RC2-128                                                                    |
|                        | - RC4-40                                                                     |
|                        | - RC4-128                                                                    |

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local-Side Certificate</strong> Required field. Specifies the certificate to be used for AS2 message decryption. This must be a reference to a certificate with a private key that was previously imported into FlowForce Server, see Configuring AS2 Certificates. In FlowForce, such objects appear with the type &quot;certificate + private key&quot;, like the second in the image below:</td>
<td></td>
</tr>
</tbody>
</table>
### Security Settings | Signature Creation

This group of settings must be defined if your organization should sign AS2 messages sent to this partner.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Algorithm         | Required field. Specifies the hash algorithm for computing the signature MIC (message integrity check). Valid values:  
                      - MD5  
                      - SHA-1 [Default]  
                      - SHA-224  
                      - SHA-256  
                      - SHA-384  
                      - SHA-512 |

<table>
<thead>
<tr>
<th>Local Side Certificate</th>
<th>Required field. Specifies the certificate issued by your organization for signing AS2 messages and MDNs sent to this partner. This must be a reference to a certificate with a private key that was previously imported into FlowForce Server, see <a href="#">Configuring AS2 Certificates</a>. In FlowForce, such objects appear with the type &quot;certificate + private key&quot;, like the second in the image below:</th>
</tr>
</thead>
</table>

### Security Settings | Signature Verification

This group of settings must be defined if your organization should verify the signature of MDNs sent by partner.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Algorithms | Required field. Specifies the algorithm(s) that should be used to compute the signed message hash in signature. If the trading partner does not use one of the algorithms below then FlowForce Server will return an MDN with an error text like: "automatic-action/MDN-sent-automatically ; failed / error: insufficient-message-  

---

[Configuring AS2 Certificates](#)
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>security</td>
<td>Also, the message will not be accepted and processed in this case. Valid values:</td>
</tr>
<tr>
<td></td>
<td>• MD5 [Default]</td>
</tr>
<tr>
<td></td>
<td>• SHA-1 [Default]</td>
</tr>
<tr>
<td></td>
<td>• SHA-224 [Default]</td>
</tr>
<tr>
<td></td>
<td>• SHA-256 [Default]</td>
</tr>
<tr>
<td></td>
<td>• SHA-384 [Default]</td>
</tr>
<tr>
<td></td>
<td>• SHA-512 [Default]</td>
</tr>
</tbody>
</table>

| Partner Certificate   | Conditional field. Specifies the certificate to be used for verifying the signature of messages and MDNs sent by partner. This must be a public certificate that you received from your trading partner and then imported into FlowForce Server, see Configuring AS2 Certificates.
|                       | If the Request Signed MDN check box is enabled, then this field must be set also. |

### Message Disposition Notification

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request MDN</td>
<td>The option <strong>Synchronous</strong> means that FlowForce will request that the partner send a synchronous MDN in reply to the AS2 message. To request no MDN, click <strong>Delete</strong> and remove this block of options.</td>
</tr>
<tr>
<td></td>
<td>Note: Asynchronous MDNs are currently not supported, see Limitations.</td>
</tr>
<tr>
<td>Request signed MDN</td>
<td>Optional field. Select this check box to request a signed MDN from the trading partner, see Message Disposition Notification.</td>
</tr>
</tbody>
</table>

### Interoperability Settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compress Data</td>
<td>Conditional field. When <strong>Use Compression</strong> option is enabled, this option specifies if compression should occur before or after data is signed for transmission to an AS2 partner.</td>
</tr>
<tr>
<td></td>
<td>For outgoing messages, the option selected must be one that your AS2 partner supports.</td>
</tr>
<tr>
<td></td>
<td>In case of incoming messages (that is, if FlowForce Server receives messages from other partners), this option is irrelevant—FlowForce</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Server will decompress messages regardless of whether they were compressed before or after signing.</td>
</tr>
<tr>
<td>MIC Verification Algorithm</td>
<td>Conditional field. This field is applicable if the Request MDN option is set (see above). It specifies what algorithm FlowForce Server should use when verifying or computing the MIC (message integrity check) used for AS2 MDN (see also RFC 4130 §7.3.1).</td>
</tr>
<tr>
<td></td>
<td>For interoperability reasons, you may need to choose Use algorithm of MDN signature if the AS2 partner runs Microsoft BizTalk. Choose Use algorithm of original message signature if the AS2 partner runs mendelson AS2.</td>
</tr>
<tr>
<td></td>
<td>When both communicating AS2 servers run FlowForce Server, this option must be identical for both.</td>
</tr>
<tr>
<td></td>
<td>The value of this field can also make a difference when an algorithm other than SHA-1 is used for signature MIC in AS2 message or in MDN, (SHA-256, for example).</td>
</tr>
</tbody>
</table>
11.7 Sending AS2 Messages

A job that sends an AS2 message to a remote partner is similar to any other FlowForce Server jobs. Namely, it can take parameters, contain various execution steps, be triggered as a scheduled job or on demand, and so on. This example shows you how to create a simple AS2 job that sends an EDIFACT file to an AS2 server.

Prerequisites

- An AS2 server must be available and configured to accept AS2 messages from HTTP clients (in this case, FlowForce Server acts as HTTP client to the remote server).
- The remote partner details must be added in FlowForce Server, see Configuring AS2 Partners. At the minimum, for a basic connectivity test, you could define a partner without any certificates (if it accepts unencrypted and unsigned connections). In this case, all you need to know is the partner's URL, the partner AS2 name, and your organization's AS2 name to communicate with this partner.

Creating the job

Create a new FlowForce Server job in the standard way (click Create | Create Job inside any container, see also Creating Jobs). Next, add an execution step that calls the /system/as2/send function. To quickly search for this function, click inside the Execute function box, and start typing the function name, for example:

![Execution Steps](image)

After you add the function to the job, its structure is loaded into the page, and fields for all the required parameters become available. To ensure the AS2 transmission is sent correctly, set the parameters as follows:

- **Partner** - This field must be a reference to a partner object configured earlier, see Configuring AS2 Partners. Click inside the field to browse for the partner object.
- **Message** - This field must contain a FlowForce expression that opens the stream you want to include in the message. For example, to send an EDIFACT file found at C:\as2\orders.edi, with a Content-Type header application/EDIFACT, enter the following expression:

  stream-open("c:\as2\orders.edi", "application/EDIFACT")

For more information about expressions in FlowForce, see The FlowForce Expression Language. The source file (be it EDI or XML) could also be a file generated with MapForce (for example, by a
previous execution step which runs a mapping that was previously deployed to FlowForce Server), see AS2 Integration with MapForce and MapForce Server.

- **Message ID** - This field must provide the value for the Message-ID header field, as a string. To generate this value, call the new-message-id expression function, as shown below.
- **Abort on error** - A job may consist of various execution steps, not just the one that is sending the AS2 message. For example, you may want to define other execution steps after the current one, in order process the MDN returned by the partner in some way. Set this parameter to TRUE (enabled) to abandon further job execution if the current execution step fails. If the Abort on error parameter is TRUE (enabled) and the current execution step fails, any subsequent execution steps will no longer be run, and the entire job will be aborted, see also Processing Steps Sequentially.

The image below illustrates a sample execution step that refers to a partner “APOLLO” and supplies an EDIFACT file in the message body with the help of a FlowForce Server expression:

As stated above, a FlowForce job may be configured to run on demand, or as a scheduled job. For information about various job triggers that can be configured, see Managing Triggers. In this example, we will configure the AS2 job to run on demand from the browser as a Web service, as shown below. Observe the name of the Web service, it is “sendAS2” in this example, but could be a different name if so required. For more information, see Exposing Jobs as Web Services.

Finally, before attempting to save the job, enter the credentials to the operating system account that FlowForce Server must run as (note these are not the same credentials as the ones you use to log on to FlowForce Server). In this example, credentials are entered directly inside the job; however, it is also possible to store them separately as a credential record, and conveniently select (refer to) them from within jobs, see also Credentials.
Now you can save the job by clicking the **Save** button at the bottom of the page.

FlowForce Server performs data integrity checks that will prevent you from saving the job if it is not configured properly. It is often the case that errors are caused by incorrect expressions supplied as parameter values, see *Handling Data Types in Steps*[^154]. If you are new to FlowForce Server, refer to the *FlowForce Expression Language*[^158] and *Job Configuration Examples*[^168] sections.

**Note:** If you need to create multiple similar jobs, be aware that FlowForce jobs can be easily duplicated, helping you save time, see *Duplicating jobs*[^167].

**Running the job**

Since this job was exposed as a Web service, you can run it by typing the Web service URL in the browser's address bar. The Web service URL is composed of the URL at which FlowForce Server service runs (for example, `http://localhost:4646/`), plus the `service/sendAS2` part, where `sendAS2` is the name of the Web service we gave previously. The final URL is therefore: `http://localhost:4646/service/sendAS2`. If you configured the FlowForce Server service to run on a different host and port, make sure to adjust this URL accordingly, see *Defining the Network Settings*[^169]. The image below illustrates the result of a successful execution as it could appear in the browser:

[^154]: Handling Data Types in Steps
[^158]: FlowForce Expression Language
[^168]: Job Configuration Examples
[^167]: Duplicating jobs
[^169]: Defining the Network Settings
The job execution results can also be viewed through the FlowForce Server log, see Viewing the Job Log.

**Processing the AS2 job result**

You have seen above how to create a simple job that consists of only one execution step which calls the `/system/as2/send` function. However, in a real life scenario, it is likely that your FlowForce Server job will need more steps.

Importantly, the return type of the `/system/as2/send` function is an **AS2 MDN** object. In order to extract useful information from this object, it must be further processed by means of FlowForce expression functions. For example, to get the message ID of the original AS2 message, you could add an execution step like the one illustrated below:
In the job above, the second step gets the original AS2 message ID as a string, by taking the result of the first step (declared as output1) as parameter. To achieve this, it calls the /system/compute function which is the usual way in FlowForce to compute an expression. The expression applies the as2-message-id expression function to the result of the first execution step (output1).

Note that FlowForce Server includes other expression functions that could be handy in various circumstances. For example, in order to determine if the AS2 call was successful, you could call the as2-success function, in a similar way as shown above. Likewise, to obtain the HTTP status of the AS2 call, you could call the as2-http-status expression function. All available expression functions are listed in the Expression Functions chapter. The ones applicable to AS2 and MIME are listed in the AS2 Expression Functions and MIME Expression Functions chapters, respectively.

An important rule when working with FlowForce expressions is to pay special attention to the return data type of each function. The data type must be compatible across all calling functions and steps; otherwise, the job cannot be saved because of validation errors. It is therefore strongly recommended that you have a basic understanding of FlowForce expressions before using them, see The FlowForce Expression Language.
11.8 Receiving AS2 Messages

With FlowForce Server, you can create jobs to receive AS2 messages from your organization's partners, process this data, and store it locally. In general, such jobs share the same characteristics as other FlowForce jobs, and, in addition, provide the following extra functionality:

- You can create, directly from the job configuration page, an AS2 service that listens to requests.
- As further illustrated below, the job that receives AS2 data takes two predefined parameters, **partner** and **message**. These parameters provide information about the sending partner and the incoming message, respectively.

Exposing a job as AS2 service roughly works in the same way as exposing a job as a Web service, see also Exposing Jobs as Web Services. Namely, the AS2 service URL is in a format like:`http(s)://<flowforce-server>:<port>/service/<as2-service-name>`, where:

- `<http(s)>` refers to the protocol that you can choose, HTTP or HTTPS (this is configured from the FlowForce Server setup page, see Defining the Network Settings).
- `<flowforce-server>` is the host name or IP address of the machine where FlowForce Server runs.
- `<port>` is the port name (by default, 4646). Note that HTTP and HTTPS have different port numbers, as configured from the setup page, and, specifically, from the "FlowForce Server" section, see Defining the Network Settings.
- **service**—this URL part is always the same and cannot be changed.
- `<as2-service-name>` is the custom name you want to give to your AS2 service. You can define this URL part when you create the job.

Depending on your needs, you can configure FlowForce Server to accept requests from unauthenticated clients (thus making the service public) or request basic HTTP authentication from clients. To make the AS2 service accessible without authentication, create the AS2 service job in a FlowForce Server container where the user **anonymous** has the following permission: "Service: Use". For more information about containers and permissions, see Permissions and Containers. For an example of such configuration, see Example: Full AS2 Message Exchange (Simple).

Prerequisites

Before you can receive AS2 data from partners, the following prerequisites must be met:

- The details of each partner from whom you will be receiving data must be added to FlowForce Server, see Configuring AS2 Partners.
- The "FlowForce Server" service must accept connections from remote clients on the designated URL, as mentioned above.

By default, FlowForce Server accepts connections from localhost on port 4646. To make the AS2 service accessible to machines other than localhost, open the setup page, and change the **Bind address** of FlowForce Server to **All interfaces (0.0.0.0)** or to a specific interface, see Defining the Network Settings. In addition, make sure that FlowForce Server is allowed to communicate through the operating system's firewall.
Note: The “FlowForce Server” service should not be confused with the “FlowForce Web Server” service. The latter is used to access the Web administration interface, accepts connections on port 8082 and has separate configuration, see also How It Works.

Creating the AS2 service

This example illustrates how to create a job that exposes an AS2 service. First, log on to FlowForce Web administration interface (see Logging on to FlowForce Server). You could create the AS2 service in the default public container; however, it is a good idea to create a separate container for it (because this service might need separate permissions). Click Configuration, and then click Create | Create Container.

Enter a container name (for example, "as2service"), and then click Save and go there. Next, click Create | Create job. The job configuration page opens:

To turn this job into an AS2 service, select the check box Make this job available at... and enter the name of the service (for example, "as2-receiver"). In addition, make sure to select AS2 service from the drop-down list.

Note that two new parameters have now been added automatically to the job:
AS2 Integration Receiving AS2 Messages

### Job Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner</td>
<td>This parameter provides information about the AS2 partner that sent the message. The parameter data type is &quot;AS2 partner&quot;. You can process this object in a subsequent step and get the partner's local or remote name as string, with the help of FlowForce expression functions as2-partner-local-name or as2-partner-remote-name.</td>
</tr>
<tr>
<td>message</td>
<td>This parameter provides access to the incoming message. The data type of the message is &quot;stream&quot;. As illustrated below, you can convert the stream to a file using FlowForce expression functions.</td>
</tr>
</tbody>
</table>

### Note:
The predefined parameters partner and message must not be deleted. If you do not use the predefined parameters in subsequent steps, you can ignore them—this does not make the job invalid. However, you will typically want to process at least the incoming message in some way (for example, save it to a file). As illustrated below, this can be done by using FlowForce expression functions, and, in particular, MIME Expression Functions. In some cases, you might want to add extra parameters to the job (for example, to define some constant value reusable across multiple steps)—if you do this, ensure that the parameter has a default value; otherwise, the job will not be started when an AS2 message arrives, and an error message will be logged.

So far, the job is configured to accept AS2 data, but it does not do anything with that data yet. In order to read the message content from the stream and save it to a file, let's add a new execution step to the job. Click **New Execution Step**, and browse for the `/system/filesystem/copy` function. Then fill the **Source** and **Target** parameters as illustrated below:
Execution Steps

The execution step above calls the `/system/filesystem/copy` function to copy data from Source to Target. Source is a FlowForce expression. In this example, the expression

\[
\text{(as-file(message))}
\]

reads the message parameter mentioned earlier and converts it to a filename, with the help of the `as-file` expression function.

The expression

\[
\text{(substring(current-message-id(), 1, -1))}
\]

does the following:

1. It gets the value of the Message-ID header field as a string, with the help of the `current-message-id` expression function. For example, a typical Message-ID could look like `<20180309125433018954-56c8aeb2fb4b478eb02f6f57662607da@somehostname>`.
2. It strips the first and last characters of the resulting string, with the help of the `substring` expression function. This makes the Message-ID look like `20180309125433018954-56c8aeb2fb4b478eb02f6f57662607da@somehostname` (notice the angle brackets "<" and ">" have now been stripped).

Finally, the string ".msg" is appended to the expression and this creates the path where FlowForce should save the incoming AS2 message. Note that the path is relative to the working directory `C:\temp`. Essentially, whenever someone will send an AS2 message to `http://<flowforce-server>:<port>/service/as2-receiver`, this job will read the message content and save it to a path like `C:\temp\20180309125433018954-56c8aeb2fb4b478eb02f6f57662607da@somehostname.msg`.

Remarks:

- The Overwrite check is not selected, meaning that the job will return an error in the event that a job with the same message ID arrives twice.
• The **Abort on error** setting is enabled, meaning that the job will fail if the copy function fails. A failed job will cause FlowForce to send a negative MDN to the partner. In this case, this option is intentionally enabled, meaning that, if FlowForce fails to save the message, it will send a negative MDN to the partner.

You have now finished creating a basic AS2 service which listens to AS2 requests and stores incoming AS2 messages locally. For an example of how this AS2 service can be consumed by clients, see [Example: Full AS2 Message Exchange](#).

In a real-life scenario, for more advanced processing, it is likely that you will need to add more execution steps to the job, and make use of other expression functions available in FlowForce. For reference to all FlowForce functions that you can call in execution steps, see [Built-in Functions](#). For a basic introduction to FlowForce expressions, refer to [The FlowForce Expression Language](#) chapter.
11.9 Example: Full AS2 Message Exchange (Simple)

This example illustrates how to configure a complete AS2 message exchange between two AS2 partners, from a FlowForce Server perspective. In this example, both the sending AS2 partner and the receiving AS2 partner are FlowForce Server instances.

Let’s call the sending server “Hermes” and the receiving server “Apollo”. Let’s also note that Hermes runs on CentOS while Apollo runs on Windows (this detail is important only for paths and firewall configuration, as shown below). The goal of this example is as follows:

- The sending server (Hermes) must successfully send an AS2 message to the receiving AS2 server (Apollo).
- The receiving server (Apollo) must successfully process the incoming message and store it locally.

This example illustrates the simplest possible communication scenario between two AS2 partners (the first permutation out of twelve possible permutations according to section 2.4.2 of RFC 4130), which essentially means the following:

- The sender sends unencrypted AS2 data
- The sender sends unsigned AS2 data
- The sender does not require that an MDN be returned in reply to the message

Other assumptions:

- Apollo and Hermes are both running on a local private network.
- The receiving AS2 server (Apollo) will accept HTTP requests from unauthenticated clients (that is, the AS2 service will be accessible publicly).

Prerequisites

- FlowForce Server Advanced Edition must be installed and licensed on both Apollo and Hermes machines.
- On both Apollo and Hermes servers, the FlowForce Web administration interface must be up and running on the configured host and port (for example, http://apollo:8082 and http://hermes:8082, assuming that “apollo” and “hermes” are the respective host names). See also Defining the Network Settings.

Configuring the sending AS2 server (“Hermes”)

1. Log on to the FlowForce Web administration interface and create a new AS2 partner called “APOLLO” (see also Configuring AS2 Partners). This partner identifies the server that will receive AS2 messages. Since encryption, signing, and MDN are not required in this simple example, the only partner settings that must be defined are as follows:
As illustrated above, the AS2 partner's name used for AS2 communication is "Apollo", while the partner object name stored in FlowForce Server is "APOLLO". The "Request URL" value assumes that the partner's host name is also apollo. If the host name is different, adjust the URL accordingly. We will configure the actual AS2 service behind this URL in a subsequent step.

2. Create a new job that sends an AS2 message.
   
a) Open to the public container, and click Create | Create job.
Enter a job name (for example, "send-as2"), and, optionally, a description.

b) Click **New filesystem trigger** and set the trigger settings as shown below. If the directory `/home/altova/as2/outgoing` does not exist on Hermes machine, create it.

As soon as you add the trigger, a parameter called `triggerfile` is added to the job. This parameter represents the file name that will trigger this job automatically, whenever you copy a file to `/home/altova/as2/outgoing`. For more information, see [File System Triggers](#).

c) Add an execution step that sends an EDI file from the local path defined previously to the AS2 partner. For more information about what this step does, see [Sending AS2 Messages](#).
d) Finally, add the credentials of the user account on the local machine (typically, the username and password that you use to log on to this machine). Note that these credentials are not the same as the username and password to the FlowForce Web administration interface. For more information, see Credentials.

e) Click Save. The job should now appear under “Active Triggers” in the FlowForce Server home page.

Configuring the receiving AS2 server (“Apollo”)

1. Configure FlowForce Server to accept connections from AS2 clients on the designated URL. In this example, AS2 clients will connect to Apollo through plain HTTP on default port 4646, so the configuration page should look as follows (see also Defining the Network Settings):
2. Make sure that FlowForce Server is allowed to communicate through the operating system's firewall. In this example, since the “Apollo” FlowForce Server runs on Windows, it must be allowed to communicate through Windows Defender Firewall.

3. Create a new FlowForce Server container; let's call it “as2service”. (In FlowForce, permissions are set at container level, so it is advisable that you create a separate container for the job that will receive AS2 messages. This way, you will be able to set AS2-specific permissions only to the required container, without affecting the permissions applicable to other existing FlowForce jobs).
4. Open the "as2service" container defined previously and create the sending partner, Hermes, as shown below. The "Request URL" value assumes that the partner's host name is also hermes. If the host name is different, adjust the URL accordingly.

```
Partner name: HERMES
Partner description:

Partner Settings
AS2 Name: Hermes

Local Side Settings
AS2 Name: Apollo

AS2 Service Settings
Receive messages: ☑️

HTTP Endpoint Settings
Request URL: http://hermes:4646/service/as2-receiver
Redirect Mode: Arbitrary redirection
Use Chunked Transfer Encoding: ☐
HTTP Authentication Credential: +
Timeout: +

Make sure that the Interoperability Settings are the same on both servers, for example:

```
Interoperability Settings
Compress Data: Before signing
MIC Verification Algorithm: Use algorithm of original message signature
```

5. Open the "as2service" container defined previously and create a new job. The purpose of this job is to expose an AS2 service that listens to AS2 requests. When a new AS2 message is received, this job will copy it to a temporary folder.
a) Select the **Make this job available via HTTP at...** check box and give a name to the AS2 service (in this example, "as2-receiver").

b) As illustrated above, select the option **AS2 service** from the drop-down list. As a result, two input parameters are added to the job, **partner** and **message**. These can be used to process and store information about the sending partner and the message, respectively. In this example, we will store the message only, as shown in a subsequent step.

c) Add an execution step that copies the received message to a local path. The FlowForce Server expressions used below essentially convert the message to a file, and compose the file name based on the **Message-ID** header field. For a more detailed explanation about these expressions, see **Receiving AS2 Messages**.

---

**Create job in /public/as2service**

- **Job name**: receive-as2
- **Job description**: Receives AS2 messages.

---

**Service**

- **Make this job available via HTTP at URL** http://<FlowForce server>/service/
- **AS2 service**

---

**Job Input Parameters**

- **Name**: partner, **Type**: AS2 partner
- **Name**: message, **Type**: stream
Make sure that the directory \C:\as2\incoming exists. This is the directory where received AS2 communications will be saved.

d) Finally, add the credentials of the user account on the local machine (typically, the username and password that you use to log on to this machine). Note that these credentials are not the same as the username and password to the FlowForce Web administration interface. For more information, see Credentials.

6. Go to the container **public / as2service**, and click Permissions. Click Add Permissions and assign the permission "Service: Use" to user **anonymous** on the "as2service" container.

The container permissions now look as follows:
This effectively makes the AS2 service public and enables anyone to access and consume it, without authentication.

**Sending the AS2 message**

On Hermes machine, copy an .edi file to the directory configured previously, `/home/altova/as2/outgoing`. When the directory polling interval elapses (60 seconds, by default), the trigger is executed, and the job sends the file to the AS2 service on Apollo machine.

To view the job result, check the FlowForce Server log, see Viewing the Job Log. If the job fails, the reason will be indicated in the log. There could be multiple reasons why a job may fail, including the following:

- The path to the EDI file on Hermes is incorrect
- The Hermes operating system credentials specified in the job are incorrect
- The Apollo service `http://apollo:4646/service/as2-receiver` is not available because the firewall on Apollo machine blocks it
- The FlowForce Server container permissions for service `http://apollo:4646/service/as2-receiver` forbid anonymous access (that is, the AS2 service is not accessible to clients)
- The "Request URL" parameter of the Apollo partner is incorrect (on Hermes machine, on Apollo machine, or both)
- The "Interoperability Settings" parameters are misconfigured for Hermes partner on Apollo machine.

On success, the receiving job on Apollo machine processes the incoming message and creates a new file at the following path: `C:\as2\incoming`. 

<table>
<thead>
<tr>
<th>User or Role name</th>
<th>Service</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous</td>
<td>Service: Use</td>
<td>Container: Read, Write inherited from /public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration: Read, Write inherited from /public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Credential: Use inherited from /public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Service: Use inherited from /public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function: Use inherited from /public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Certificate: Use inherited from /public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AS2 Partner: Use inherited from /public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Security: Read inherited from /</td>
</tr>
<tr>
<td>authenticated</td>
<td>Service: Use</td>
<td>Container: Read, Write inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration: Read, Write inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Credential: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Service: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Certificate: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AS2 Partner: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Security: Read, Write inherited from / and authenticated</td>
</tr>
<tr>
<td>root</td>
<td>Service: Use</td>
<td>Container: Read, Write inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration: Read, Write inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Credential: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Service: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Certificate: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AS2 Partner: Use inherited from /public and authenticated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Security: Read, Write inherited from / and authenticated</td>
</tr>
</tbody>
</table>
11.10  Example: Full AS2 Message Exchange (Advanced)

This example illustrates a more advanced AS2 message exchange, with encryption and signing, between two AS2 partners that both run FlowForce Server. Before you follow this tutorial, make sure that you have already followed the previous one, which covers the basics, see Example: Full AS2 Message Exchange (Simple).

This example illustrates the most complex communication scenario between two AS2 partners (the twelfth permutation out of twelve possible permutations according to section 2.4.2 of RFC 4130), which essentially means the following:

- The sender sends encrypted AS2 data
- The sender sends signed AS2 data
- The sender requests that the receiver returns a signed MDN in reply to the message

Assumptions

- The same sender and receiver are used as in the previous example, respectively: Hermes (FlowForce Server on Linux) and Apollo (FlowForce Server on Windows)
- Hermes wants to send to Apollo an encrypted and signed message, and requires a signed MDN in return
- Apollo and Hermes are both running on a local private network.
- The receiving AS2 server (Apollo) will accept HTTP requests from unauthenticated clients (that is, the AS2 service will be accessible publicly).

Prerequisites

- FlowForce Server Advanced Edition must be installed and licensed on both Apollo and Hermes machines.
- On both Apollo and Hermes servers, the FlowForce Web administration interface must be up and running on the configured host and port (for example, http://apollo:8082 and http://hermes:8082, assuming that "apollo" and "hermes" are the respective host names). See also Defining the Network Settings.

Set up Apollo's certificates

In this configuration step, the following takes place:

1. Apollo generates a public certificate and a private key and imports both into FlowForce Server.
2. Apollo sends the public certificate (without the private key) to Hermes.

Why this is necessary:

- Before sending the message to Apollo, Hermes needs Apollo's public key to encrypt it. Upon receiving the message from Hermes, Apollo will decrypt it using his own private key.
- Before sending the MDN requested by Hermes, Apollo will sign it using his own private key. Upon receiving the signed MDN, Hermes needs Apollo's public certificate to verify the signature.

For the scope of this example, we will generate a self-signed certificate using the OpenSSL library (https://www.openssl.org/) included with Cygwin (https://cygwin.com). This is for demo purposes only; in a real
life scenario, you might want to use other tools to generate the SSL certificate, or you might have it already available in your organization.

To generate the self-signed certificate for Apollo, open the Cygwin terminal and type the following:

```
openssl req -x509 -newkey rsa:2048 -keyout apollo_private.pem -out apollo_public.pem -days 365
```

When prompted to enter a pass phrase, type the password under which you would like to encrypt the private key, and remember it. You will later need this password to import the certificate into FlowForce Server. Go through all wizard steps, and enter all the required fields ("Country", "State or Province Name", "Locality Name", "Organization Name", "Department Name", "Common Name", and "Email").

When you finish the wizard, the command above generates two files, `apollo_private.pem`, and `apollo_public.pem`, in Cygwin's home directory (for example, `C:\cygwin64\home<user>`), if you installed Cygwin to `C:\cygwin64`. Because this pair can only be uploaded as one single file into FlowForce Server, run the following additional command to copy the public certificate into the private key file:

```
cat apollo_public.pem >> apollo_private.pem
```

On the Apollo machine, log on to FlowForce Server, click the Configuration menu, and then click Create > Create Certificate.
Enter the certificate name and description, click Browse and select the **apollo_private.pem** file create previously. Make sure to enter the password that you created earlier in this step, and click **Save**.

The public+private certificate pair is now imported into Apollo's FlowForce Server. Notice that the icon and descriptive text indicate that this certificate file contains both:

To send the public key to Hermes, copy the **apollo_public.pem** file to Hermes machine. Next, log on to FlowForce Server on Hermes machine and import it using the same steps as above (this time a private key is not present in the file, so no password is necessary).
Notice that the icon and descriptive text indicate that this certificate file contains only the public certificate (no private key).

Notice that the icon and descriptive text indicate that this certificate file contains only the public certificate (no private key).

Set up Hermes’s certificates
In this configuration step, the following takes place:

1. Hermes generates a public certificate and a private key and imports it into FlowForce Server
2. Hermes sends the public certificate (without the private key) to Apollo
3. Apollo imports Hermes’s public certificate into FlowForce Server

Why this is necessary:

- Before sending the message to Apollo, Hermes will sign it using his own private key.
- Upon receiving the message from Hermes, Apollo will verify the signature of the message using Hermes’s public certificate.
First, create Hermes's public certificate and private key, following the same steps as for Apollo. Be sure to replace the file names:

```
openssl req -x509 -newkey rsa:2048 -keyout hermes_private.pem -out hermes_public.pem -days 365
```

In addition, the "Organization name", "Common Name", etc. must be those of Hermes:

```
altova@DOC-W10x64 ~
$ openssl req -x509 -newkey rsa:2048 -keyout hermes_private.pem -out hermes_public.pem -days 365
Generating a 2048 bit RSA private key
...........................................+++******+++writing new private key to 'hermes_private.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:-----You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
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]:Hermes
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:hermes
Email Address []:hermes@example.org
```

Next, combine both files into a single one using the command:

```
cat hermes_public.pem >> hermes_private.pem
```

Next, import `hermes_private.pem` into FlowForce Server on Hermes machine:
Next, copy `hermes_public.pem` to Apollo machine and import it into FlowForce Server:

Enable AS2 encryption, signing, and MDN signature verification on Hermes
On Hermes machine, edit the APOLLO partner settings as follows:
Enable AS2 decryption, MDN signing, and signature verification on Apollo
On Apollo machine, edit the HERMES partner settings as follows:
According to the requirements stated above, Hermes requires that Apollo send an MDN to acknowledge the AS2 transmission. We can compute the status of the incoming MDN (success, failure) with the help of `as2-success` expression function. To achieve this, log on to FlowForce on Hermes machine, and open the "send-as2" job created previously in Example: Full AS2 Message Exchange (Simple). Next, modify the job as shown below:

**Security Settings**

- **Encryption**
  - Decryption Algorithms: DES, 3DES, AES-128, AES-192, AES-256, RC2-40, RC2-64, RC2-128
  - Local Side Certificate: /public/ApolloPrivate

- **Signature Creation**
  - Algorithm: SHA-1
  - Local Side Certificate: /public/ApolloPrivate

- **Signature Verification**
  - Algorithms: MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
  - Partner Certificate: /public/HermesPublic

**Message Disposition Notification**

- Request MDN

**Interoperability Settings**

- Compress Data: Before signing
- MIC Verification Algorithm: Use algorithm of original message signature

---

**Process the MDN**

According to the requirements stated above, Hermes requires that Apollo send an MDN to acknowledge the AS2 transmission. We can compute the status of the incoming MDN (success, failure) with the help of `as2-success` expression function. To achieve this, log on to FlowForce on Hermes machine, and open the "send-as2" job created previously in Example: Full AS2 Message Exchange (Simple). Next, modify the job as shown below:
Send the AS2 message
You are now ready to send the encrypted and signed AS2 message from Hermes to Apollo. On Hermes machine, copy an .edi file to the directory configured previously `/home/altova/as2/outgoing`. When the directory polling interval elapses (60 seconds, by default), the trigger is executed, and the job sends the file to the AS2 service on Apollo machine. The directory `C:\as2\incoming` on Apollo machine should now contain the message sent by Hermes, for example:
To see if the job has failed or has executed successfully, check the system's log (you may need to do this not only on Hermes, but also on the Apollo machine). For more information, see Viewing the Job Log.

The log contains information about any errors that may occur in relation to this transmission. For example, if Hermes sends unencrypted data but Apollo expects it to be encrypted, then the job fails and a corresponding message is logged.
# 12 Command Line Interface

FlowForce provides a command line interface which can be used for administration purposes (such as licensing, troubleshooting, and internal database backup). The command line interface includes the following commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td>Displays help for the command supplied as argument.</td>
</tr>
<tr>
<td>assignlicense</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>compact db</td>
<td>Reduces the size of FlowForce .db files if they contain deleted records.</td>
</tr>
<tr>
<td>createdb</td>
<td>Creates a new FlowForce database.</td>
</tr>
<tr>
<td>debug</td>
<td>Starts the application in debug mode.</td>
</tr>
<tr>
<td>exportresourcestrings</td>
<td>Exports all application resource strings to an XML file</td>
</tr>
<tr>
<td>foreground</td>
<td>Starts the application in foreground mode.</td>
</tr>
<tr>
<td>initdb</td>
<td>Creates or updates the FlowForce database.</td>
</tr>
<tr>
<td>install</td>
<td>Installs the application as a Windows service.</td>
</tr>
<tr>
<td>licenseserver</td>
<td>Registers FlowForce Server with the Altova LicenseServer on the local network.</td>
</tr>
<tr>
<td>migratedb</td>
<td>Migrates FlowForce Server data from a previous version to the latest version.</td>
</tr>
<tr>
<td>repair</td>
<td>Starts the application in repair mode.</td>
</tr>
<tr>
<td>resetpassword</td>
<td>Resets the password of the root user to the default value, and grants to the root user all privileges.</td>
</tr>
<tr>
<td>setdeflang</td>
<td>sdil</td>
</tr>
<tr>
<td>start</td>
<td>Starts the application as a service.</td>
</tr>
<tr>
<td>uninstall</td>
<td>Uninstalls the application as a Windows service.</td>
</tr>
<tr>
<td>upgradedb</td>
<td>Upgrades the FlowForce Server database to the latest version.</td>
</tr>
<tr>
<td>verifylicense</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>
</tr>
</tbody>
</table>
Location of the FlowForce Server executable

<table>
<thead>
<tr>
<th>Platform</th>
<th>Description</th>
</tr>
</thead>
</table>
| Linux    | On Linux systems, the FlowForce Server executable (flowforceserver) is located by default at:  
|          | /opt/Altova/FlowForceServer2020/bin/flowforceserver  
|          | where the first forward slash indicates the root directory. |
| macOS    | On macOS systems, the FlowForce Server executable (flowforceserver) is located by default at:  
|          | /usr/local/Altova/FlowForceServer2020/bin |
| Windows  | On Windows systems, the FlowForce Server executable (FlowForceServer.exe) is located by default at:  
|          | <ProgramFilesFolder>\Altova\FlowForceServer\bin\FlowFo  
|          | rceServer.exe |

General syntax

The general command line syntax for FlowForceServer is as follows:

Windows

```bash
FlowForceServer --h | --help | --version | <command> [options] [arguments]
```

UNIX

```bash
flowforceserver --h | --help | --version | <command> [options] [arguments]
```

where

- `--h | --help` Displays the help text.
- `--version` Displays the version of FlowForce Server.

The options and arguments in square brackets are optional.

**Note:** On Unix systems, when you are inside the directory of the FlowForce Server executable, you might need to prepend the current directory 

```bash
./flowforceserver --version
```

macOS

```bash
./FlowForceServer --version
```
12.1 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
```
12.2 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.
12.3 compactdb

Purpose
Reduces the size of FlowForce .db files if they contain deleted records. This command is particularly useful after running the \texttt{archive-log} or \texttt{truncate-log} system maintenance functions.

Syntax

\texttt{FlowForceServer compactdb \[options\]}

\textbf{Note:} On Linux systems, use an all-lowercase \texttt{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 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 \texttt{FlowForce Server Application Data} ).</td>
</tr>
</tbody>
</table>
12.4 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.
```
12.5 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 \texttt{CTRL+C}.

Syntax

\begin{verbatim}
FlowForceServer debug [options]
\end{verbatim}

\textbf{Note:} On Linux systems, use an all-lowercase flowforceserver to call the executable.

Options

\begin{verbatim}
--datadir=VALUE
\end{verbatim}

\textit{VALUE} is the path of the data directory.
12.6 **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
```
12.7 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.
12.8 initdb

Purpose
Creates 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

FlowForceServer initdb [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.
12.9 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.
12.10 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 |
12.11 **migratedb**

**Purpose**
Migrates FlowForce Server data from a previous version to the latest version.

**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**

```
"C:\Program Files (x86)\Altova\FlowForceServer2020\bin\FlowForceServer.exe" migratedb --datadir=C:\ProgramData\Altova\FlowForceServer2020\data --olddatadir=C:\ProgramData\Altova\FlowForceServer2017\data
```
12.12   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\FlowForceServer2020\data
```
12.13 resetpassword

**Purpose**
Resets the password of the root user to the default value, and grants to the 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]
```

**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 resetpassword --datadir=C:\ProgramData\Altova\FlowForceServer2020\data
```
12.14  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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>English</td>
</tr>
<tr>
<td>es</td>
<td>Spanish</td>
</tr>
<tr>
<td>de</td>
<td>German</td>
</tr>
<tr>
<td>fr</td>
<td>French</td>
</tr>
<tr>
<td>ja</td>
<td>Japanese</td>
</tr>
</tbody>
</table>

Example

```
FlowForceServer setdeflang de
```
12.15 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.
12.16 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.
12.17 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\FlowForceServer2020\data
12.18 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

|--l, --license-key=VALUE

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.
13  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/as2
- /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

As you create jobs, you will typically need to refer to file paths on the machine where FlowForce Server runs, or file paths on the network. When referring to a Windows network path, such as 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 syntax is in the form:

```
\server\sharedfolder\filepath
```

Where:

- server references the server name in the network (defined by the DNS)
- sharedfolder references a label defined by the administrator (for example, admin$ is generally the root directory of the operating system installation).
- filepath refers to the subdirectories below the share.
13.1 /system

The /system container includes all the FlowForce built-in functions. Only the abort, compute, and compute-string functions are found directly under this container. Other functions are distributed in sub-containers, according to their area of applicability (for example, AS2 functions, file system functions, mail functions, and so on).

13.1.1 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.
13.1.2 compute

Full path: /system/compute
Built-in Functions

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 The FlowForce Expression Language.</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

See also Creating a "Hello, World" Job.
13.1.3 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".
13.2 /system/as2

The /system/as2 container includes the `send` function used to send an AS2 message to an AS2 partner.

### 13.2.1 send

**Full path:** /system/as2/send

Sends an AS2 message to a remote AS2 server. In order to call this function from a job, the AS2 partner's details (including any applicable certificates) must be already configured in FlowForce Server. See also [Creating the AS2 Job](#).

This function returns an **AS2 MDN** object which encapsulates the actual MDN returned by the server and auxiliary information from protocol. To get additional information from the **AS2 MDN** object (for example the HTTP status, or the MDN of the original message), add an execution step that calls the required **AS2 expression functions**.

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partner</td>
<td>AS2 Partner</td>
<td>References the “AS2 partner” object, see <a href="#">Configuring AS2 Partners</a>.</td>
</tr>
<tr>
<td>Message</td>
<td>stream</td>
<td>The content of the AS2 message to send, as a stream object. The stream required by this field can be converted from a file (for example, XML or EDI file) by means of a FlowForce Expression, for example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stream-open(&quot;C:\files\myfile.edi&quot;, &quot;application/EDIFACT&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Notice that the</strong> <code>stream-open</code> **function above also supplies the message <strong>Content-Type</strong> header as second parameter. Other values for <strong>Content-Type</strong> can also be used if necessary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For an introduction to expressions in FlowForce, see <a href="#">The FlowForce Expression Language</a>.</td>
</tr>
<tr>
<td>Abort on error</td>
<td>Boolean</td>
<td>This Boolean parameter determines what should be the</td>
</tr>
</tbody>
</table>
### Built-in Functions

**Name** | **Type** | **Description**
---|---|---

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 the following examples:

- [Example: Full AS2 Message Exchange (Simple)](#)
- [Example: Full AS2 Message Exchange (Advanced)](#)
13.3  /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.

13.3.1  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</td>
</tr>
</tbody>
</table>
### 13.3.2 delete

**Full path:** `/system/filesystem/delete`

Deletes a file from the path supplied as argument.

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, 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>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 <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**

See Copy Files.
### 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

#### 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 <code>c:\dir1\dir2\dir3</code> 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, <code>c:somedirectory</code>). 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.

13.3.4 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>Source</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>Destination</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>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 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:\somédirectory). If</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
|            |                     | relative paths are used, they will be resolved against the working directory.

### 13.3.5 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>
13.4 /system/ftp

The /system/ftp container includes functions used to perform FTP operations on a remote FTP server.

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 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.

13.4.1 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,</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>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>
<tr>
<td>Target file</td>
<td>string</td>
<td>The name of the file that you want delete from the server. 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>
13.4.2 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 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>Target directory</td>
<td>string</td>
<td>The name of the directory that you want to create.</td>
</tr>
</tbody>
</table>
### Aborting on Error

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 <code>Abort on error</code> is FALSE, the function will return Boolean FALSE as well. If <code>Abort on error</code> is TRUE, the job execution is aborted. The default value is TRUE.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

### 13.4.3 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.</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</td>
</tr>
</tbody>
</table>
## Built-in Functions

### Altova FlowForce Server 2020 Advanced Edition

<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>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>the file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optional parameter.</td>
<td></td>
<td></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>Source file</td>
<td>string</td>
<td>The name of the file that you want to move. Mandatory parameter.</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. 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>

### 13.4.4 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>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 where you want to retrieve 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>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.</td>
</tr>
</tbody>
</table>
### Built-in Functions

#### Overwrite target
- **Name:** Overwrite target
- **Type:** boolean
- **Description:** When true, causes the destination file to be overwritten. The default value is false.

#### Abort on error
- **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.

#### Working directory
- **Name:** Working directory
- **Type:** string
- **Description:** 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
- **Name:** Account
- **Type:** string
- **Description:** The FTP account name of the user allowed access to the files on the remote server.

### 13.4.5 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>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 where you want to delete the directory.</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>
<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>
</tbody>
</table>
13.4.6  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. 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 store 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>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</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>
<tr>
<td>Target file</td>
<td>string</td>
<td>The name the file should have once it is uploaded to the FTP Server. This can be different from the Source File. 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>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>
<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>

**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.

### 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>

### Execution Steps

**Execute function** `/system/ftp/store`

- **FTP Server**: `10.100.63.200`
- **Port**: `21`
- **Directory on host**: `uploads`
- **Login credentials**: `Select existing credential: /public/my.ftp.credentials`
- **Use passive mode**: 
- **Source file**: `{filename-with-extension(triggerfile)}`
- **Target file**: `{filename-with-extension(triggerfile)}`
- **Abort on error**: 
- **Working directory**: `C:\FlowForce\Upload`
- **Account**: 

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.
13.5  /system/mail

The /system/mail container includes the functions send and send-mime, used to send email.

13.5.1  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.

Note: Before using this function, ensure 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>
<tbody>
<tr>
<td>From</td>
<td>string</td>
<td>Email address from which the e-mail message is to be sent, for example: flowforce@&lt;hostname&gt;.</td>
</tr>
<tr>
<td>To</td>
<td>string</td>
<td>Recipient's email address. Mandatory parameter.</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>string</td>
<td>Body text of the message, as string. The message body supports ASCII as well as Unicode characters. Optional parameter.</td>
</tr>
<tr>
<td>Attachment</td>
<td>string as file</td>
<td>File name of the attachment sent with the email.</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</td>
</tr>
</tbody>
</table>


### Examples
See Adding Error Handling to a Job.

#### 13.5.2 send-mime

Full path: `/system/mail/send-mime`

Sends e-mail to the specified recipients, generally the administrator.

Note: Before using this function, ensure 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: flowforce@&lt;hostname&gt;.</td>
</tr>
<tr>
<td>To</td>
<td>string</td>
<td>Recipient's email address. Mandatory parameter.</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 stream expression functions to create streams from strings or files. Call MIME expression functions to 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 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>
</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 body of the message. 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.

The "Message body" field contains a FlowForce expression. The expression converts the result of the first execution step (of type string) to a stream. The `stream-from-string` function was called for that purpose. The encoding and the desired MIME type were passed as arguments to the function.

The image attachment was taken from a local path, `C:\sample.png`. 
Note that the image above was cut in order to fit on a print page. The "Attachment" field contains the following expression:

```
add-mime-header(stream-open('C:\sample.png', 'image/png'), 'Content-Disposition', 'attachment; filename="sample.png")
```

The expression above does the following:

- The `stream-open` function opens the image as a stream
- The `add-mime-header` function adds the "Content-Disposition" header to the stream. This way, the image attachment will have the intended name ("sample.png").
13.6 /system/maintenance

The /system/maintenance container includes functions used to perform maintenance operations on the server.

13.6.1 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>

13.6.2 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.

### 13.6.3  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>
13.7 /system/shell

The /system/shell container includes the function commandline, used to execute shell commands or scripts.

13.7.1 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 behaviour 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>
<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>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><strong>string as directory</strong></td>
<td>Specifies the working directory of the job (for example, <code>c:\somedirectory</code>). If relative paths are used, they will be resolved against the working directory.</td>
</tr>
</tbody>
</table>

**Examples**

The following job executes a Windows batch file called **DoTransform.bat**. Assuming that the **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 `C:\codegen\xslt2`.

The following job calls RaptorXML Server to run an XSLT transformation with parameters. It is assumed that the PATH environment variable contains the path to the RaptorXML Server executable, for example `C:\Program Files (x86)\Altova\RaptorXMLServer2020\bin`. For more information about RaptorXML Server, see [https://www.altova.com/raptorxml](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](Check if a path exists).
14 Expression Functions

This chapter provides reference to the FlowForce expression functions. For conceptual information that will help you understand how to use expressions, see The FlowForce Expression Language.

The expression functions are grouped as shown below:

- Step Result Functions 334
- Stream Functions 335
- File System Functions 345
- File Path Functions 345
- List Functions 346
- String Functions 355
- Boolean Functions 363
- Runtime Information Functions 365
- AS2 Expression Functions 369
- MIME Expression Functions 370
14.1 Step Result Functions

Step result functions allow you to process the result returned by jobs (or the result returned by execution steps within jobs).

14.1.1 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.

### 14.1.2 failed-step

Returns the step result. Using this function is meaningful when you are handling errors with protected blocks. 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.

As such, this function does not indicate the step where the error occurred. To find the result's attributes, pass this function as argument to functions such as stdout or stderr, for example:

```plaintext
stderr(failed-step())
stdout(failed-step())
```

**Signature**

failed-step() -> result

**Examples**

See the following examples:

- Adding Error Handling to a Job
- Validate an XML Document with Error Logging

### 14.1.3 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</td>
</tr>
</tbody>
</table>
### Name | Type | Description
---|---|---
| | | particular value in the result.

### 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:

#### Execution Steps

In the job configuration above, the first step runs the mapping and returns the result as `mapping_result`. In the second step, the expression

\[
\text{as-file(nth(results(mapping_result), 0))}
\]

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:

```plaintext
{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:

```plaintext
{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

### 14.1.4 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**

`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](#)

14.1.5 stderr

Returns the standard error of the result. Fails if the result does not provide a standard error.

Signature

\[
\text{stderr} \left(\text{result: result}\right) \rightarrow \text{stream}
\]

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.
14.2 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.

14.2.1 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

\[
\text{as-file}(\text{stream}: \text{stream}) \rightarrow \text{string}
\]

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

```
Execute function /system/compute
Parameters:
Expression: as-file(stream-from-string("This is my file content"))

Assign this step's result to result as T0

Execute function /system/filesystem/copy
Parameters:
Source: {result}
Target: file.txt
Overwrite: ✔️
Abort on error: ✔️
Working directory: C:\FlowForce
```

See also Validate an XML Document with Error Logging.
14.2.2 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](#)

14.2.3 empty-stream

Creates an empty stream.

**Signature**

```
empty-stream() -> stream
```

14.2.4 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
```
### Stream Functions

#### 14.2.5 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 contenttype to associate to the stream. The default is <code>contenttype=text/plain</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:
14.3 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.

14.3.1 list-files

Lists the files in the path (which may terminate with a wildcard) and returns the resulting 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

\[
\text{list-files(path:string) -> list of string}
\]

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.

14.3.2 list-directories

Lists the subdirectories in the path (which may terminate with a wildcard) and returns the resulting string list.

Signature

\[
\text{list-directories(path:string) -> list of string}
\]

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>
14.3.3  read-lines

Reads the lines from the given file and returns them as a list of strings.

**Signature**

\[
\text{read-lines}(\text{filename: string, encoding: string} = \text{"UTF-8"}) \rightarrow \text{list of string}
\]

**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>
14.4  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).

14.4.1  extension

Extracts the file extension from a path.

Signature

\[
\text{extension(path:string) -> string}
\]

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":

\[
\text{extension("c:\temp\file.txt")}
\]

14.4.2  filename

Extracts the file name (without extension) from a path.

Signature

\[
\text{filename(path:string) -> string}
\]

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":

\[
\text{filename("c:\temp\file.txt")}
\]
14.4.3 filename-with-extension

Extracts the file name and extension from a path.

Signature

```plaintext
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":

```plaintext
filename-with-extension("c:\temp\file.txt")
```

14.4.4 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 `/system/filesystem/move` function and enter as Source and Destination the values shown below:

The parameter value `{triggerfile}` in the Source field instructs FlowForce to move specifically the file which triggered the mapping. The parameter value

```
{parent-directory(triggerfile)}processed
```

in the 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

```
{parent-directory(triggerfile)}
```

calls the 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 `source.xml`) into the current working directory (for example, `C:\FFSERV`).
2. The trigger fires and `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.

```
14.5 List Functions

List functions are used to create and disassemble lists. Lists always contain items of a single type (for example, only strings, only numbers, or only nested lists with the same item type); there are no mixed type lists.

14.5.1 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**

\[
\text{char}(\text{number: number}) \rightarrow \text{string}
\]

**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:
14.5.2 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:
### 14.5.3 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)
```

---

### 14.5.4 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.
14.5.5 length

Returns the number of items in the list.

**Signature**

\[
\text{length}(\text{list}: \text{list}) \rightarrow \text{number}
\]

**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>
14.5.6  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

\[
\text{list}(\text{item1: any type, item2: any type, itemN: any type}) \rightarrow \text{list}
\]

Parameters

<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:

\[
\text{list}(1,2,3)
\]

The following expression returns the list \(['a', 'b', 'c']\). All list items are of string type:

\[
\text{list}('a','b','c')
\]

14.5.7  nth

Returns the specified item from the list. The index is zero-based. Fails if the index is out of bounds.

Signature

\[
\text{nth}(\text{list: list, index: number}) \rightarrow \text{item}
\]

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>
</tbody>
</table>
### index

<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)
```

---

### slice

Returns a partial list from a list.

#### Signature

```text
slice(list: list, start: number, end: number=length(list)) -> 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)
```
14.6 String Functions

The string functions perform basic string operations, such as concatenation, extracting a substring from a string, trimming, splitting, and others.

14.6.1 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

```plaintext
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":

```plaintext
concat('a', 'b', 'c')
```

14.6.2 contains

Returns true if the first string contains at least one occurrence of substring, otherwise false.

Signature

```plaintext
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>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>substring</td>
<td>string</td>
<td>The string value to check for.</td>
</tr>
</tbody>
</table>

**Examples**
The following expression returns `true`:
```
contains('cat','a')
```
The following expression returns `false`:
```
contains('cat','b')
```

### 14.6.3 `ends-with`

Returns `true` if the string supplied in the `string` argument ends with the string supplied in the `end` argument.

**Signature**
```
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`:
```
ends-with('cat', 't')
```
The following expression returns `false`:
```
ends-with('cat', 'a')
```
14.6.4 find-all

Extracts all occurrences of pattern in the string, where 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"]

14.6.5 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:

\[
\text{number('1')}
\]
14.6.6  split

Splits the string supplied as argument at each occurrence of separator.

Signature

```plaintext
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', ';')
```

14.6.7  starts-with

Returns true if the string supplied in the string argument starts with the string supplied in the start argument.

Signature

```plaintext
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')
```

The following expression returns false:

```plaintext
starts-with('dog', 'C')
```
14.6.8        string

Computes the string representation of the given number, i.e. converts the number supplied as argument into a string.

Signature

\[ \text{string}(\text{number}: \text{number}) \rightarrow \text{string} \]

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 converted to string.</td>
</tr>
</tbody>
</table>

Examples

The following expression converts the numeric value 1 into the string "1":

\[ \text{string}(1) \]

14.6.9        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

\[ \text{string-join}(\text{list}: \text{list of string}, \text{separator}: \text{string} = \"\\") \rightarrow \text{string} \]

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:

\[\text{string-join(list('a', 'b', 'c'), ';')}\]

14.6.10  **string-length**

Returns the number of characters in the string.

**Signature**

\[\text{string-length(string:string) -> number}\]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The input string.</td>
</tr>
</tbody>
</table>

Examples

The following expression will return 3:

\[\text{string-length('cat')}\]

14.6.11  **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**

\[\text{substring(string:string, start:number, end:number) -> string}\]
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":

\[
\text{substr('FlowForce',4)}
\]

The following expression will return "t":

\[
\text{substr('Altova',2,3)}
\]

The following expression will return "ltov":

\[
\text{substr('Altova',1,-1)}
\]

14.6.12 trim

Removes leading and trailing whitespace characters from the string (Space, Tab, Line Feed, Carriage Return, Form Feed, and Vertical Tab).

Signature

\[
\text{trim(string:string) -> string}
\]

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>
14.6.13  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>

14.6.14  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>
### 14.7 Boolean Functions

The Boolean functions are used to evaluate true/false expressions.

#### 14.7.1 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>

#### 14.7.2 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>
14.7.3 false

Returns Boolean false.

Signature

false() -> Boolean

14.7.4 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 condition evaluates to true.</td>
</tr>
<tr>
<td>valueFalse</td>
<td>any type</td>
<td>Specifies a subexpression to return when condition evaluates to false.</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")
```
14.7.5  **not**

Returns the negation of the Boolean value supplied as argument.

**Signature**

\[
\text{not}(\text{value}: \text{Boolean}) \rightarrow \text{Boolean}
\]

**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>

14.7.6  **true**

Returns Boolean **true**.

**Signature**

\[
\text{true}() \rightarrow \text{Boolean}
\]
14.8  Runtime Information Functions

The runtime information functions can be used to handle the details of the currently running jobs.

14.8.1  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

\[ \text{instance-id}() \rightarrow \text{string} \]

14.8.2  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 Changing the Logging Settings.

Signature

\[ \text{log(expression:T0)} \rightarrow \text{string} \]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>T0</td>
<td>The FlowForce expression to be logged, of type T0 (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](image-url)
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:

<table>
<thead>
<tr>
<th>Date</th>
<th>Severity</th>
<th>Module</th>
<th>User</th>
<th>Instance</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-01-16 17:39:27</td>
<td>INFO</td>
<td>flowforce</td>
<td>root</td>
<td>17</td>
<td>Executing FlowForce.assign with parameters: Expression: list('C:\source\File 01.txt', 'C:\source\File 02.txt', 'C:\source\File 03.txt', 'C:\source\File 04.txt', 'C:\source\File 05.txt', 'C:\source\File 06.txt', 'C:\source\File 07.txt', 'C:\source\File 08.txt', 'C:\source\File 09.txt', 'C:\source\File 10.txt', '...')</td>
</tr>
</tbody>
</table>

To prevent truncation from happening, enclose the expression inside the `log` function, and save the job configuration.

### Execution Steps

- **Execute function** `/system/compute`
- **Parameters:**
  - **Expression:** `log(list-files('C:\source'))`

```
Assign this step's result to name as T0
```

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 \texttt{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 \texttt{log} function, so FlowForce will hide the \texttt{log} function next time when you open the job configuration page. The difference between the \texttt{log} button and the \texttt{log} function is that the former logs the entire expression displayed in the text box, whereas the \texttt{log} function can be used selectively for smaller sub-expressions, for example:

\begin{verbatim}
Execute function /system/compute

Parameters:

Expression: contains(\texttt{["haystack"], "needle"})

Assign this step's result to name as T0
\end{verbatim}

### 14.8.3 \texttt{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**

\texttt{slot-number()} \rightarrow \texttt{number}
14.9 AS2 Expression Functions

The AS2 expression functions are applicable to jobs that send AS2 messages to remote servers, see AS2 Integration.

14.9.1 as2-disposition

Extracts the disposition header value from the MDN returned by the /as2/send function. The header value will be returned as originally received, unless transmission failed, in which case a synthetic failure notification is returned. Example of disposition value:

automatic-action/MDN-sent-automatically; processed/error: decryption-failed

Signature

as2-disposition(result: AS2 MDN) -> string

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>AS2 MDN</td>
<td>A value of type AS2 MDN.</td>
</tr>
</tbody>
</table>

14.9.2 as2-http-status

Extracts the HTTP status from the MDN returned by the /as2/send function. The HTTP status will be in the 200 range for successful MDNs. Failed MDNs might contain a different status when failure was at the HTTP level, or contain 0 when no HTTP response was received.

Signature

as2-http-status(result: AS2 MDN) -> number

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>AS2 MDN</td>
<td>A value of type AS2 MDN.</td>
</tr>
</tbody>
</table>
14.9.3 as2-mdn-serialize

Returns the MDN as a stream so that it can be serialized (further processed or stored somewhere).

**Signature**

\[
\text{as2-mdn-serialize}(\text{result: AS2 MDN}) \rightarrow \text{stream}
\]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>AS2 MDN</td>
<td>A value of type AS2 MDN.</td>
</tr>
</tbody>
</table>

14.9.4 as2-message-id

Extracts the message ID from the MDN returned by the `/as2/send` function. Note this ID is not the same as the message ID of the MDN. For failed MDNs, the message ID may be an empty string. This function may be useful for logging.

**Signature**

\[
\text{as2-message-id}(\text{result: AS2 MDN}) \rightarrow \text{string}
\]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>AS2 MDN</td>
<td>A value of type AS2 MDN.</td>
</tr>
</tbody>
</table>

14.9.5 as2-partner-local-name

Returns the name of the local AS2 partner as configured in the given AS2 Partner object, see Configuring AS2 Partners. The local AS2 partner is FlowForce Server.

**Signature**

\[
\text{as2-partner-local-name}(\text{partner: AS2 Partner}) \rightarrow \text{string}
\]

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner</td>
<td>AS2 Partner</td>
<td>A value of type AS2 Partner.</td>
</tr>
</tbody>
</table>
14.9.6 as2-partner-remote-name

Returns the name of the remote AS2 partner as configured in the given AS2 Partner object, see Configuring AS2 Partners.

Signature

\[
\text{as2-partner-remote-name} \left( \text{partner:AS2 Partner} \right) \rightarrow \text{string}
\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner</td>
<td>AS2 Partner</td>
<td>A value of type AS2 Partner.</td>
</tr>
</tbody>
</table>

14.9.7 as2-success

Returns true if the MDN indicates successful transmission.

Transmission is successful if HTTP transmission succeeds, the MDN can be verified against its signature (if enabled), and the MDN indicates success. When "Abort on error" is turned on for /as2/send then it is unnecessary to use this function.

Signature

\[
\text{as2-success} \left( \text{result:AS2 MDN} \right) \rightarrow \text{Boolean}
\]

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>AS2 MDN</td>
<td>A value of type AS2 MDN.</td>
</tr>
</tbody>
</table>

14.9.8 as2-signed

Returns true if the MDN was signed and the signature verified successfully.

Transmissions that failed at the HTTP layer are never signed correctly. This function is unnecessary when:

a. "Abort on error" is enabled for /as2/send, and
b. "Request signed MDN" option was enabled for the AS2 partner, see Configuring AS2 Partners.

Signature

\[
\text{as2-signed} \left( \text{result:AS2 MDN} \right) \rightarrow \text{Boolean}
\]
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>AS2_MDN</td>
<td>A value of type AS2_MDN.</td>
</tr>
</tbody>
</table>
14.10 MIME Expression Functions

The MIME expression functions are applicable to jobs that need to manipulate the headers of MIME (Multipurpose Internet Mail Extensions) entities.

14.10.1 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>stream</td>
<td>The stream to which the header should be added.</td>
</tr>
<tr>
<td>key</td>
<td>string</td>
<td>The key from the key-value pair.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>The value from the key-value pair.</td>
</tr>
</tbody>
</table>
14.10.2  add-mime-headers

Returns a stream with all headers from headers added.

Signature

\[
\text{add-mime-headers}(s: \text{stream}, \text{headers}: \text{list of (string, string)}) \rightarrow \text{stream}
\]

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>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.

\[
\text{add-mime-headers}(\text{empty-stream()}, \text{list}((\text{Content-Disposition}, \text{attachment; name=something}), (\text{Content-Transfer-Encoding}, \text{7bit})))
\]

14.10.3  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

\[
\text{current-message-id()} \rightarrow \text{string}
\]
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`.

```plaintext
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.

```plaintext
C:\temp\{nth(split(substring(current-message-id(), 1, -1), '@'), 0)}.msg
```

14.10.4 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

```plaintext
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 `msg` contains the header `Content-Disposition: attachment; filename="GETMSG.edi"`, the following expression will return "GETMSG.edi":

```plaintext
mime-get-content-disposition-param(msg, "filename")
```
14.10.5  get-mime-content-id

Returns the value of the Content-ID header from the stream supplied as argument, if such header exists; otherwise, it returns the value of the default argument.

Signature

```plaintext
get-mime-content-id(s:stream, 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>default</td>
<td>string</td>
<td>Specifies the value to return when Content-ID header does not exist. By default, this is an empty string.</td>
</tr>
</tbody>
</table>

Examples

Let's suppose that stream `msg` has the header `Content-ID: <root.attachment>`. The expression

```plaintext
get-mime-content-id(msg, "")
```

returns "<root.attachment>" in this case. If no such header exists, the expression above returns an empty string (the value of the second argument).

14.10.6  get-mime-content-type-param

Returns the parameter param from the "Content-Type" 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 Multiple Attachments (MA). Namely, it can extract the starting document Content-ID and Content-Type specified as parameters 'start' and 'type' to 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>
</tbody>
</table>
Expression Functions

MIME Expression Functions

### Name | Type | Description
--- | --- | ---
param | string | Specifies the name of the parameter to return.
default | string | Specifies the value to return when the requested param does not exist. By default, this is an empty string.

### Examples
Assuming that stream `msg` contains the header `Content-Type: text/html; charset=utf-8`, the following expression will return "utf-8":

```plaintext
mime-get-content-type-param(msg, "charset", "ascii")
```

#### 14.10.7 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 key 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 `msg` contains the header `Content-Disposition: attachment; filename="GETMSG.edi"`, the following expression will return `attachment; filename="GETMSG.edi"`:

```plaintext
get-mime-header(msg, "Content-Disposition", "")
```

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).
14.10.8 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 headers parameter to the add-mime-headers expression function.

Signature

get-mime-headers(s:stream) -> list of (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>
</tbody>
</table>

14.10.9 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 default argument.

Signature

get-stream-filename(stream:stream, default:string="") -> 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 input stream.</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>

14.10.10 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>

### 14.10.11 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:

```
Match ::= Single ("," Single)*
Single ::= Spaces? Type-Match ( Spaces? ";" Spaces? Parameter )* Spaces?
Type-Match ::= 
            "/*" |
            Type "/*" |
            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>
<tr>
<td>accept</td>
<td>string</td>
<td>Specifies the custom-defined accept rules.</td>
</tr>
</tbody>
</table>

### Examples

The following expression will return `true` if stream `msg` contains the header `Content-Type: text/html; charset=utf-8` or `Content-Type: text/plain; charset=utf-8`.

```plaintext
is-mime-content-type(msg, "text/*; charset="utf-8")
```
The following expression will return true if stream \textit{msg} contains the header \textbf{Content-Type: application/rss+xml} or \textbf{Content-Type: application/svg+xml}.

\begin{verbatim}
is-mime-content-type(msg, "application/*+xml")
\end{verbatim}

You can also match multiple rules by separating them with a comma. For example, the following expression will return true if stream \textit{msg} contains the header \textbf{Content-Type: text/xml} or \textbf{Content-Type: application/xml}:

\begin{verbatim}
is-mime-content-type(msg, "text/xml, application/xml")
\end{verbatim}

### 14.10.12 	exttt{mime-content-encode}

Applies \textit{encoding} as \textbf{Content-Transfer-Encoding} to stream \textit{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 \textbf{Content-Transfer-Encoding} and re-encodes it using the specified encoding. The new \textbf{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**

\begin{verbatim}
mime-content-encode(s:stream, encoding:string="") -> stream
\end{verbatim}

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{s}</td>
<td>\textit{stream}</td>
<td>Specifies the input stream.</td>
</tr>
<tr>
<td>\textit{encoding}</td>
<td>\textit{string}</td>
<td>Specifies the encoding to apply. By default, this is an empty string.</td>
</tr>
</tbody>
</table>

### 14.10.13 	exttt{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

\[ \text{mime-flatten}(s:\text{stream}) \rightarrow \text{stream} \]

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>
</tbody>
</table>

14.10.14 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

\[ \text{mime-multipart}(\text{subtype}: \text{string}, s:\text{stream}) \rightarrow \text{stream} \]

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>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"))
```
14.10.15 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.

```
mime-multipart-related(list(part1, part2))
```
14.10.16 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>

14.10.17 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.
Signature

\[
mime-parse(s:stream) \rightarrow stream
\]

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>

14.10.18 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.

Signature

\[
mime-split-multipart(s:stream) \rightarrow list of stream
\]

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>

14.10.19 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:

'\('<' UTC timestamp '-' random hex value 32 characters long '0' host name related text '>'\)

For example: <20180306154822808383-5933b654b26c4495bb0b619ab72b3bc6@myservername>.

Signature

\[
new-message-id() \rightarrow string
\]
14.10.20 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>

14.10.21 set-mime-content-disposition

Sets the parameter of a MIME "Content-Disposition" header found in stream s.

**Note for FlowForce Server Advanced Edition users:** This function is useful when sending AS2 messages with the optional AS2 profile FileName preservation (FN). See also the function `get-mime-content-disposition-param` for reading the file name.

**Signature**

```
set-mime-content-disposition(s: stream, disposition: string, filename: 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>disposition</td>
<td>string</td>
<td>The disposition value of the &quot;Content-Disposition&quot; header to set.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>The filename value of the &quot;Content-Disposition&quot; header to set. By default, this is an empty string.</td>
</tr>
</tbody>
</table>

**Examples**

The following expression sets the "Content-Disposition" header as follows: `Content-Disposition: attachment; filename="GETMSG.edi"`: 
set-mime-content-disposition(msg, "attachment", "GETMSG.edi")

14.10.22 set-mime-content-id

Returns a stream with the "Content-ID" header set to value, and all other headers and content untouched. You can also achieve the same result using the set-mime-header function; this function represents a more direct approach.

Signature

set-mime-content-id(s:stream, value: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 string.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Specifies the value to set in the &quot;Content-Disposition&quot;.</td>
</tr>
</tbody>
</table>

Examples

Let's assume that you want to set the value of the "Content-ID" header in stream msg to <root.attachment>. To do this, use the following expression:

set-mime-content-id(msg, "<root.attachment>")

14.10.23 set-mime-header

Returns a stream with header key set to value, and all other headers and content untouched. If you need to change several headers at once, you might want to use the set-mime-headers function.

Signature

set-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>stream</td>
<td>Specifies the input stream.</td>
</tr>
</tbody>
</table>
### MIME Expression Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td>Specifies the key of the header to set.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Specifies the header value to set.</td>
</tr>
</tbody>
</table>

#### Examples

To override the "Content-Type" header, use:

```plaintext
set-mime-header(s, "Content-Type", "text/plain; charset=iso-8859-1")
```

### 14.10.24 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"}))
```
15 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>content()</td>
<td>Web service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· /system/compute-string</td>
<td>stdout()</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>trim()</td>
<td></td>
</tr>
<tr>
<td>Copy Files</td>
<td></td>
<td>· /system/filesystem/copy</td>
<td>list-files()</td>
<td>Web service</td>
</tr>
<tr>
<td>Create a Job from a MapForce Mapping</td>
<td>· MapForce mapping</td>
<td></td>
<td>-</td>
<td>Timer</td>
</tr>
<tr>
<td>Use a Job as Step of Another Job</td>
<td>· /system/filesystem/copy</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Create a Directory Polling Job</td>
<td>· MapForce mapping</td>
<td></td>
<td>-</td>
<td>File system</td>
</tr>
<tr>
<td></td>
<td>· /system/filesystem/move</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Add Error Handling to a Job</td>
<td>· /system/shell/commandline</td>
<td>failed-step()</td>
<td>exitcode()</td>
<td>Timer</td>
</tr>
<tr>
<td></td>
<td>· /system/compute</td>
<td></td>
<td>stdout()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· /system/mail/send</td>
<td></td>
<td>stderr()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· /system/compute-string</td>
<td></td>
<td>content()</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instance-id()</td>
<td></td>
</tr>
<tr>
<td>Expose a Job as a Web Service</td>
<td>· MapForce mapping</td>
<td></td>
<td>-</td>
<td>Web service</td>
</tr>
<tr>
<td>Cache Job Results</td>
<td>· /system/shell/commandline</td>
<td>stdout()</td>
<td>-</td>
<td>Web service</td>
</tr>
<tr>
<td></td>
<td>· /system/compute</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Create a Job from a StyleVision Transformation</td>
<td>· StyleVision transformation</td>
<td>results()</td>
<td>nth()</td>
<td>Timer</td>
</tr>
<tr>
<td></td>
<td>· /system/compute</td>
<td></td>
<td>as-file()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· /system/filesystem/copy</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Validate a Document with RaptorXML</td>
<td>· /RaptorXML/valany</td>
<td></td>
<td>-</td>
<td>Timer</td>
</tr>
<tr>
<td>Validate an XML Document with Error Logging</td>
<td>· /RaptorXML/valxml-withxsd</td>
<td>failed-step()</td>
<td>stdout()</td>
<td>Web service</td>
</tr>
<tr>
<td></td>
<td>· /system/compute</td>
<td></td>
<td>stderr()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· /system/filesystem/copy</td>
<td></td>
<td>as-file()</td>
<td></td>
</tr>
<tr>
<td>Use RaptorXML to Pass Key/Value Parameter Pairs</td>
<td>· /RaptorXML/xslt</td>
<td>list()</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Generate Multiple PDFs from Multiple XMLs</td>
<td>· MapForce mapping</td>
<td>as-file()</td>
<td>results()</td>
<td>Web service</td>
</tr>
<tr>
<td></td>
<td>· StyleVision transformation</td>
<td></td>
<td>filename()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>· /system/compute</td>
<td></td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
15.1 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
- FlowForce Server must be running at the configured network address and port (see Setting the 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 (this example uses the default /public container).
2. Click Create, and then select Create job.
3. 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.
5. Declare the execution result as `string`.

6. Select the Make this job available via HTTP... check box and type "HelloWorld" as service name (see Exposing Jobs as Web Services).

7. Under "Credentials", select an existing credential record or specify a local credential (see Credentials).

8. 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 test the job, enter `http://127.0.0.1:4646/service/HelloWorld` in the browser's address bar, provided that FlowForce Server runs at the default host and port name. If you have defined other host and port settings in the Configuration page (see Setting the Network Address and Port), change the address accordingly. If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

If you use Internet Explorer to test FlowForce Server jobs exposed as Web services, you may need to disable the "Show friendly HTTP error messages" option in the Advanced tab.

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 log of the job in FlowForce Server to identify the error (see Viewing the Job Log).
15.2 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
- FlowForce Server must be running at the configured network address and port (see Setting the 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 (in this example, we are using /public).
2. Click Create, and then select Create job.
3. Add a job name (CheckPath, in this example) and, optionally, a job description.

![Create job in /public](image)

4. Under Job Input Parameters, click +, and add the parameter path, as shown below.

![Job Input Parameters](image)

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 `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 `/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 `/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 `string`. 
10. Under Service, click to select the **Make this job available via HTTP** check box, and enter **CheckPath** as name of the service (see Exposing Jobs as Web Services).

11. Under “Credentials”, select an existing credential record or specify a local credential (see Credentials).

12. Click **Save**.

### Running the job

To test the job, enter **http://127.0.0.1:4646/service/CheckPath** in the browser’s address bar, provided that FlowForce Server runs at the default host and port name. If you have defined other host and port settings in the Configuration page (see Setting the Network Address and Port), change the address accordingly. If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

If you use Internet Explorer to test FlowForce Server jobs exposed as Web services, you may need to disable the "Show friendly HTTP error messages" option in the **Advanced** tab.

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).
15.3 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:\Source to a new directory C:\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
- FlowForce Server must be running at the configured network address and port (see Setting the 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:\Source to directory C:\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 Administration Interface, and click Configuration. Open the /public container and 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:\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 [The FlowForce Expression Language](#).

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 `list-files("C:\Source\*.*")` 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, enter `http://127.0.0.1:4646/service/CopyFilesService` in the browser's address bar, provided that FlowForce Server runs at the default host and port name. If you have defined other host and port settings in the Configuration page, change the address accordingly, as described in **Setting the Network Address and Port**. If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

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:

![Log View](image_url)

15.4 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. Deploying a mapping means that MapForce organizes the resources used by the mapping into an object and passes it to FlowForce Server. Once the mapping is deployed to FlowForce Server, you can create a server job from it. This particular example instructs you how to configure the mapping job to run daily at a specific time.

Prerequisites

- FlowForce Server is running at the configured network address and port (see Setting the 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:\temp directory).

Demo files used

- CompletePO.mfd, available at the following path: 
  <Documents>\Altova\MapForce2020\MapForceExamples.

Creating the job

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".

3. On the File menu, click **Deploy to FlowForce Server**.

4. Enter the server name and port of the Web administration interface in the respective fields (for example, **localhost** and **8082**, if FlowForce Server is running on the same machine at the default host and port).

5. Enter your FlowForce Server user name and password, and, next to **Login**, 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. Make sure the **Open web browser to create new job** check box is selected. Although you can change the path where the mapping should be deployed, for the purpose of this example, leave the selected path as is.

7. Click **OK**. When deployment finishes, the FlowForce Server Administration Interface opens in your web browser, and a partially filled in job page is displayed. As shown below, an execution step is created automatically with some pre-filled parameters. For the scope of this example, leave the parameter values as is.
8. In the **Working-directory** box, enter the path to a directory where FlowForce Server must save the job output. This example uses C:\temp 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.

9. Under Triggers, click **new Timer**.
10. 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:
11. Under Credentials, select an existing credential record or specify a local credential (see also Credentials 157).

12. Click Save. 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 working directory specified previously. To see whether the job executed successfully, refer to the job log (see Viewing the Job Log 157).
15.5  **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 `CompletePO.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 generated file from the temporary folder to an archive folder (we will call this job `copy2archive`).
2. Modify the `CompletePO.job` to include the `copy2archive` job as an additional execution step.

**Prerequisites**

- FlowForce Server is running at the configured network address and port (see Setting the 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:\temp` and `C:\archive` directory).
- Complete the steps described in the Creating a Job from a MapForce Mapping example.

**Demo files used**

- `CompletePO.job` available in the `/public` container of FlowForce Server administration page.

**Creating the job**

1. Click Configuration, and then navigate to the `/public` container.
2. Click Create, and then select Create Job.
3. Enter the name of the job (in this example, "copy2archive").
4. Under Execution steps, add the first execution step, as follows:
   a. Next to Execute function, navigate to the `system/filesystem/copy` function (for further information about functions, see Built-in Functions).
   b. In the Source text box, type the path of the file to be copied (for example, "C:\temp\CompletePO.xml").
   c. In the Target text box, type the destination path (for example, "C:\archive"). This must be an existing directory on the operating system where FlowForce Server runs. If you want to rename the file when it is copied, add the file name to the path, for example "C:\archive\PurchaseOrders.xml".
   d. Click to select the Overwrite check box. This instructs FlowForce Server to overwrite any file with the same name found at the destination path.
   e. Leave the Abort on error option as is.

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...
Use a Job as Step of Another Job

is aborted. The default value is TRUE.

f. In the Working directory box, type the working directory (for example, "C:\temp").

![Execution Steps]

5. Under credentials, select an existing credential record or specify a local credential (see Credentials).

6. Click Save.

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.

To add the "copy2archive" job as a step of "CompletePO" job:

1. In the /public container, click to open the CompletePO.job record.
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 location of the copy2archive job. Now the execution steps look as follows:

![Execution Steps]
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 from the temporary location to the archive location. To see whether the job executed successfully, refer to the job log (see Viewing the Job Log).
15.6 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

- FlowForce Server is running at the configured network address and port (see Setting the 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:\temp and C:\archive directory).

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\MapForce2020\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.

This mapping essentially 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.39

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 applicable to this step. This example uses C:\temp 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:\temp\*.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 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:\temp\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):
   - Execute function: /system/filesystem/move
   - Source: c:\temp\ShortInfo.xml
   - Destination: c:\archive
   - Working directory: c:\temp

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 (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 (see Viewing the Job Log).
15.7 Add Error Handling to a Job

This example shows how to add error handling to a simple job that lists the contents of a directory. More specifically, it shows you how to configure FlowForce Server as follows:

- Whenever 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
- FlowForce Server is running at the configured network address and port (see Setting the Network Address and Port)
- The FlowForce Server mail settings have been configured (see Setting the Mail Parameters)
- 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).

Tips

- This example uses FlowForce Server expressions, which are required to handle the job return values. To understand this example better, you might want to have a look at the instance-id, stderr, stdout, and failed-step expression functions.
- 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 machine where FlowForce Server runs, create a directory where the job output will be saved. This can be, for example, C:\FlowForce, or a similar path. Note that your user account must have read and write rights for this directory.
2. Log in to the FlowForce Server Web administration interface, open the /public container, and then click Create Job. Next, enter a name and, optionally, a description for the Web service you are creating.
4. 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 /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>
</tbody>
</table>
On Windows, this command lists recursively the contents of a directory. If the directory is not specified like in this case, then it lists the contents of the working directory (see below).

<table>
<thead>
<tr>
<th>Abort on error</th>
<th>Leave this option as is.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working directory</td>
<td>Enter the path to the working directory created previously, for example C:\FlowForce</td>
</tr>
</tbody>
</table>

5. 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 <code>/system/mail/send</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Enter the email address of the sender, for example flowforce@localhost. Leave this field empty if you have configured the mail settings from the administration page.</td>
</tr>
<tr>
<td>To</td>
<td>Enter your email address.</td>
</tr>
<tr>
<td>Subject</td>
<td>Enter the subject of the notification email as follows:</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Job {instance-id()} has failed" /></td>
</tr>
<tr>
<td></td>
<td>The part between curly braces is a FlowForce expression which calls the <code>instance-id</code> function to get the unique ID of the current (failed) job instance.</td>
</tr>
<tr>
<td>Message body</td>
<td>Type the following:</td>
</tr>
<tr>
<td></td>
<td>![Exit Code: {string(exitcode(failed-step()))} Standard Error: {content(stderr(failed-step()))}] (image)</td>
</tr>
<tr>
<td></td>
<td>The parts between curly braces are two FlowForce expressions. These expressions essentially get the erroneous output and convert it to a string that will be the body of the email:</td>
</tr>
<tr>
<td></td>
<td>- The <code>failed-step</code> 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 <code>exitcode</code> or <code>stderr</code> functions, see below.</td>
</tr>
<tr>
<td></td>
<td>- The <code>exitcode</code> function gets the actual exit code of the error from the result, as a number. Recall that this step is inside an On Error handler, so we expect that there is an exit error code (and possibly also error output). Had this been an On Success handler or a standard step, result would need to be processed in a different way (for example, as described in Create a Job from a StyleVision Transformation).</td>
</tr>
<tr>
<td></td>
<td>- The <code>stderr</code> function gets the standard error output of the error from the result, as a stream.</td>
</tr>
</tbody>
</table>
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 converts the error output from a stream to a string.

6. Click **new error/success handler**, and then select **Always**.

7. 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 (for example, <code>C:\FlowForce</code>).</td>
</tr>
</tbody>
</table>

At this stage, the job should look as follows (provided you did not use different paths or shell commands).
8. Under “Triggers”, add a Timer trigger which runs every N minutes (for example, 2 minutes). Make sure to adjust the date, time, and time zone as required.
9. Under "Credentials", select an existing credential record or specify a local credential (see Credentials).

10. Click **Save**.

At this stage, you have completed the job configuration, and you might want to do one of the following:

- To test the "Always" condition, wait for the trigger condition to be met. Whenever the trigger condition is met, a new job ID is appended to the contents of the **JobLog.txt** file.
- To test the "On Error" condition, change the parameters of the first step to some deliberately incorrect value (perhaps, by specifying a path that does not exist). In this case, FlowForce Server will send an email to the address specified in the recipient field of the "On Error" handler. Additionally, it will log the job ID in the **JobLog.txt** file, since this behaviour has been configured to happen in the "Always" condition.
- To see whether the job executed successfully, refer to the job log (see Viewing the Job Log).
15.8 Expose a Job as a Web Service

This example illustrates how to expose a sample mapping job as a Web service. The sample mapping was already designed with MapForce; it reads data from a Microsoft Access database which stores a list of person records. The mapping retrieves from the database only person records whose last name begins with a specific letter (supplied as a parameter). You will learn how to deploy the existing mapping from MapForce to FlowForce Server (either on the same or on a different machine), and turn it into a Web service. After completing this example, you will be able to invoke the Web service from a browser.

Prerequisites

- FlowForce Server is running at the configured network address and port (see Setting the 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)

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\MapForce2020\MapForceExamples.

- DB_PhoneList.mfd (the MapForce mapping design file)
- altova.mdb (the Microsoft Access database from which the mapping reads data).

Preparing the mapping for deployment to a different machine

Since this mapping reads data from a database file, additional configuration must be done before deploying the mapping, as explained in this section. If MapForce and FlowForce Server are installed on the same computer, you can skip to section "Deploying the mapping" below.

**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.

Before attempting to deploy the mapping to the target machine, do the following:

1. On the target machine, configure the "FlowForce Web Server" service to listen either on all interfaces, or on a specific IP address other than the local host, see Defining the Network Settings. You can check whether the service is configured correctly by accessing the following URL from the browser: http://<FlowForce Web Server><port>. Make sure that the incoming connections to the specified address and port are not blocked by the firewall.

2. As mentioned above, the job created in this example must be available as a Web service. In FlowForce, any requests to jobs exposed as Web services are handled by the "FlowForce Server" service (not by the "FlowForce Web Server" service, see also How It Works). Therefore, if the Web service should be accessible to HTTP clients outside of the local host, the "FlowForce Server" service must also 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: \texttt{http://<FlowForce Server><port>/service/}. All jobs that are exposed as Web services (if any) should appear as links directly in the browser window.

Before it is deployed, the mapping must also be reconfigured to use relative instead of absolute paths, as follows:

1. Open the \texttt{DB_PhoneList.mfd} mapping in MapForce, right-click the mapping area, and select \textbf{Mapping Settings} from the context menu.
2. Clear the \textbf{Make paths absolute in generated code} check box.
3. 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 Access database must be manually copied from the source machine to the target machine. Copy the \texttt{altova.mdb} database file from the directory ..
Documents\Altova\MapForce2020\MapForceExamples on the source machine to some empty directory on the target machine. In this example, the target directory is "C:\FlowForceWorkingDir". This directory 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 Mappings for Server Execution.

Deploying the mapping

1. Open the DB_PhoneList.mfd in MapForce.
2. On the File menu, click Deploy to FlowForce Server. For the purpose of this example, we assume the mapping is deployed to the default path (/public container). 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 FlowForce Server (see above).

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 web 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 container), and then click Create Job.

To configure the job:

1. Under "Job Input Parameters", create a new input parameter of type string and name it NamePrefix.
2. Under "Execution Steps", next to NamePrefix, click Set to, and then select NamePrefix. This sets the value of the mapping parameter NamePrefix to the value of the NamePrefix input parameter created in previous step.

3. Next to Working-directory, enter "C:\FlowForceWorkingDir" (this must be the same directory where the Access file was previously copied).

4. Under "Service", select the Make this job available via HTTP check box, and enter "GetPhoneList" as name of the service.

5. Under Credentials, select an existing credential record or specify a local credential (see also Credentials). Note: Do not confuse these credentials with the ones used to access the FlowForce Server Web administration interface. Also, make sure that the user entered here is able to access the altova.mdb database file from the working directory; otherwise, the job will fail to execute successfully.

6. Click Save.

Invoking the Web service
You can now invoke the Web service you just created, as follows:

1. Open a Web browser and type the following URL in the address bar (replace [FlowForceServer] and [Port] with the settings configured in the administration page):
If you use Internet Explorer to test FlowForce Server jobs exposed as Web services, you may need to disable the "Show friendly HTTP error messages" option in the **Advanced** tab.

2. When prompted to supply credentials, enter the credentials you use to access the FlowForce Server Web administration interface.

3. When prompted to enter the parameters of the Web service, enter F (assuming that you want to retrieve all persons whose surname begins with "F").

4. Click **Submit**. FlowForce Server processes the job and returns the result.

If the job executes successfully, the job's output is displayed directly in the browser (and it is also generated in the working directory `C:\FlowForceWorkingDir`). Otherwise, if you are seeing an execution error, refer to the job log for more details (see **Viewing the Job Log**).
15.9  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; however, 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 that will be called by MapForce. MapForce Enterprise Edition acts as a client that calls the Web service created with FlowForce Server. You may also use a different Web client and achieve the same result.

- FlowForce Server is running at the configured network address and port (see Setting the 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:\POST. 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
Login to the FlowForce Server Web administration interface, open the /public container, and then click Create Job. Next, enter a name and, optionally, a description for the Web service you are creating.
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.

**Job Input Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>stream</td>
<td>The body of the HTTP request</td>
</tr>
</tbody>
</table>

Next, add a new execution step and configure it as follows:

**Execution Steps**

- **Execute function**: /system/filesystem/copy
- **Parameters**:
  - **Source**: `{as-file(data)}`
  - **Target**: `file(instance-id)01.json`
  - **Working directory**: C:\POST

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 **Set to** 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:\POST** (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 "POST_JSON" or a 
similar name for the new 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

As a first step to test the Web service, you can call it from the browser, as follows:

1. Open a Web browser and type the following URL in the address bar:
   
   http(s)://[FlowForceServer]:[Port]/service/POST_JSON

   By default, this URL is http://localhost:4646/service/POST_JSON, if you did not change the FlowForce Server host and port. Otherwise, replace [FlowForceServer] and [Port] with the settings configured in the FlowForce setup page, see Defining the Network Settings. If you use Internet Explorer to test FlowForce Server jobs exposed as Web services, you may need to disable the “Show friendly HTTP error messages” option in the Advanced tab.

2. When prompted to supply credentials, enter your FlowForce Server credentials.

   This is only for testing the Web service and should not be done in 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.

3. Click Browse and select the JSON file to be submitted in the POST request.
4. 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:\POST`. 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 **Insert** menu, click **Web Service Function**. The Web Service Call Settings dialog box opens.
3. 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.

![Web Service Call Settings dialog box](image)

4. Click the **Edit** button next to "HTTP Security Settings", and enter the credentials required to access the Web service.
Note: If you would like to supply the credentials from the mapping instead of saving them in this dialog box, 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.

5. Click OK to close the dialog box. The mapping now looks as follows:

6. Add to the mapping a simple input that will supply the JSON data, by selecting the **Insert** | **Insert Input** menu command. Also, enter some sample JSON data to be used for executing this mapping at design time, like the one shown below:
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.

7. Add the output of the mapping, by selecting the **Insert | Insert Output** menu command.

8. Drag the **charset-encode** and **mime-entity** functions from the Libraries window and make the connections as shown below. You will also need to add two constants, by selecting the **Insert | Constant** menu command.

Web service call with unstructured body

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:\POST 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 [https://www.altova.com/documentation](https://www.altova.com/documentation).
15.10 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
- FlowForce Server is running at the configured network address and port (see Setting the 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).

Tip

- 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.

Creating the cache producer and the cache consumer jobs

1. Click Configuration, and then navigate to the /public container.
2. Click Create, and then select Create Job.
3. In the Job Name box, enter DirectoryListing.
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>Assign this step's result to</td>
<td>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.</td>
</tr>
</tbody>
</table>
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:</td>
</tr>
<tr>
<td></td>
<td><code>stdout(dir)</code></td>
</tr>
</tbody>
</table>

The `stdout` function converts the raw result returned by the previous execution step into a stream of data (see Step Result Functions).

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.

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 `DirectoryListingCached` as the name of the Web service.

9. Under Service, click to select the **Make this job available via HTTP** check box, and enter `DirectoryListing` as name of the service.

At this stage, the job should look as follows (provided you did not use different paths or shell commands).

10. Under Credentials, select an existing credential record or specify a local credential (see Credentials).

11. Click **Save**.
At this stage, you have completed the configuration of both the cache producer and the cache consumer jobs. To compare the performance of both jobs, you can do the following:

1. Access the URL of the cache producer service (DirectoryListing) in the browser, that is http://[FlowForceServer]:[Port]/service/DirectoryListing (if you are having trouble accessing the Web service, check the configured network address and port). 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.
2. Refer to the job log to see how long it took for the job to complete (see Viewing the Job Log).
3. Next, do the same for the cache consumer service (DirectoryListingCached). Since this service consumes the cache rather than executing the directory listing, it is expected to take significantly less time to complete.
15.11 Create a Job from a StyleVision Transformation

This example shows you how to create a FlowForce Server job from a StyleVision transformation. First, you will deploy a demo transformation file from StyleVision to FlowForce Server. Deploying a transformation file means that StyleVision organizes the resources used by the transformation into an object and passes it to FlowForce Server. Once the transformation is deployed to FlowForce Server, you will create a server job from it. 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
- FlowForce Server is running at the configured network address and port (see Setting the 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).
- On the operating system where FlowForce Server runs, the following directories must exist and you must have rights to create files in them:
  o c:\archive
  o c:\temp

Demo files used

- AutoCalc.sps, available at the following path relative to the StyleVision examples project: Examples > Basics AutoCalc.sps. (To open the StyleVision examples project in StyleVision, click Examples on the Project menu.)

Creating the job

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.
3. When prompted to save the transformation as PXF file, leave the default settings as is, and click OK.
4. Enter the server name and port of the Web administration interface in the respective fields (for example, localhost and 8082, if FlowForce Server is running on the same machine at the default port).
5. Enter your FlowForce Server user name and password, and, next to Login, 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. Make sure the **Open web browser to create new job** check box is selected. Although you can change the path where the transformation should be deployed, for the purpose of this example, leave the selected path as is.

7. Click **OK**. When deployment finishes, the FlowForce Server Administration Interface opens in your web browser, and a partially filled in job page is displayed. As shown below, an execution step is created automatically with some pre-filled parameters.

**Note:** The option **GenerateHtmlOutputAsMime** is not selected 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.

8. Next to Parameters / OutHtml, click the **button. This declares the AutoCalc.html as output file.
9. In the **Assign this step’s result** box, enter **step1**. This instructs FlowForce Server that the result returned by the first execution step of the job has the name **step1**. We will need to refer to this result subsequently.

10. In the **Working-directory** box, enter the path to a directory where FlowForce Server must save the job output. This example uses `c:\temp` as working directory.

A working directory is a parameter required by [execution steps](https://flowforceserver.com/documentation/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.

11. Under the execution step that is already created, click **new Execution step** and configure the step 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(step1), 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 <code>results</code> to get the array returned by <strong>step1</strong> 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.</td>
</tr>
<tr>
<td></td>
<td>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 function <code>as-file</code> to declare it as a file.</td>
</tr>
<tr>
<td>Assign this step's result to</td>
<td>Enter <strong>step2</strong>. This instructs FlowForce Server that the result returned by the step has the name <strong>step2</strong>. We will need to refer to this result subsequently.</td>
</tr>
</tbody>
</table>
```

12. 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 <strong>Set to</strong>, and then select <strong>step2</strong>.</td>
</tr>
<tr>
<td>Target</td>
<td><code>c:\archive\AutoCalc.html</code></td>
</tr>
<tr>
<td>Overwrite</td>
<td>Select the <strong>Overwrite</strong> check box.</td>
</tr>
<tr>
<td>Working directory</td>
<td><code>c:\temp</code></td>
</tr>
</tbody>
</table>
```

At this stage, the Execution Steps section of the job page should look as follows:
13. Under Triggers, click **new Timer**.
14. 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:

15. Under Credentials, select an existing credential record or specify a local credential (see also **Credentials**).
16. Click **Save**. 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:\archive` directory. To see whether the job executed successfully, refer to the job log (see Viewing the Job Log).
15.12 Validate a Document with RaptorXML

This example shows you how to create a job which validates an XML Schema file. The validation job uses the `valany` RaptorXML function built into FlowForce Server. 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
- FlowForce Server is running at the configured network address and port (see Setting the 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 uses the `address.xsd` file available in the RaptorXML Server installation folder. On a Windows 8 system running 32-bit FlowForce Server, the path would be `C:\Program Files (x86)\Altova\RaptorXMLServer2020\examples\address.xsd`, unless you installed RaptorXML Server in a different folder.

Creating the job

1. Click Configuration, and then click the "RaptorXML" container.
2. Click the `valany` function (you can also find this function in any container that corresponds to a specific RaptorXML release, for example 2016r2).
3. Click the Create Job button at the bottom of the page. This creates a job with the default name "valany.job". Edit the job name if necessary.
The parameters applicable to the `valany` function are now visible on the page. Note that the mandatory parameter that you need to supply is shown as an expanded field.

4. In the **File** field, enter the path and file name of the file that you want to validate, for example, C: \Program Files (x86)\Altova\RaptorXMLServer2020\examples\address.xsd.

5. Under Triggers, click **new Timer** and create a trigger that will run the job at a specific time in future (see also **Timer Triggers**).

6. Under Credentials, select an existing credential record or specify a local credential (see also **Credentials**).
7. Click **Save**. The job will run at the time specified in the trigger. To see whether the job executed successfully, refer to the job log (see Viewing the Job Log). Specifically, in the log entry details page, result="OK" means that the validation process was successful, for example:

```plaintext
file:///C:/Program%20Files%20(x86)/Altova/RaptorXMLServer/examples/address.xsd:
runtime="0ms" result="OK"
```

If the file did not validate, the log entry displays result="Fail".
15.13 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. (The RaptorXML Server functions become available in FlowForce Server when RaptorXML Server is licensed.) 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.

In this particular 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. However, if necessary, 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
- FlowForce Server is running at the configured network address and port (see Setting the 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:\FlowForce` 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\RaptorXMLServer2020\examples\NanonullOrg.xml`. On a 64-bit Windows running 32-bit FlowForce Server, the path would be `C:\Program Files (x86)\Altova\RaptorXMLServer2020\examples\NanonullOrg.xml`, unless you installed RaptorXML Server in a different folder.

Creating the job

1. Log on to FlowForce Server and navigate to a container where you have permission to create new jobs (in this example, we are using `/public`).
2. Click Create, and then select Create job.
3. Add a Job name (ValidateFile, in this example) and, optionally, a job description.

```
Create job in /public
```

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>Set the value to /RaptorXML/valxml-withxsd. You can also find this function in any container that corresponds to a specific RaptorXML release, for example /RaptorXML/2020/valxml-withxsd.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Set the value of the XML File parameter to:</td>
</tr>
</tbody>
</table>
|                  | C:\Program Files (x86)
\Altova\RaptorXMLServer2020\examples\NanonullOrg.xml                                          |
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>/system/compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Set the value of Expression to:</td>
</tr>
<tr>
<td></td>
<td>as-file(stdout(failed-step()))</td>
</tr>
<tr>
<td>Assign this step's result to</td>
<td>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.</td>
</tr>
</tbody>
</table>
7. Under the previous execution step, click the button and choose to add a new execution step, with the following settings:

<table>
<thead>
<tr>
<th>Execute function</th>
<th>/system/filesystem/copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>MyResult variable declared earlier.</td>
</tr>
<tr>
<td>Target</td>
<td>C:\FlowForce\ValidateFile.log</td>
</tr>
<tr>
<td>Overwrite</td>
<td>select the check box. 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>
</tbody>
</table>
8. Under Service, click to select the **Make this job available via HTTP** check box, and enter **ValidateFile** as name of the service.

9. Under “Credentials”, select an existing credential record or specify a local credential (see [Credentials](#)).

10. Click **Save**.

To test the job, enter [http://127.0.0.1:4646/service/ValidateFile](http://127.0.0.1:4646/service/ValidateFile) in the browser's address bar, provided that FlowForce Server runs at the default host and port name. If you have defined other host and port settings in the Configuration page (see [Setting the Network Address and Port](#)), change the address accordingly. If prompted for credentials when accessing the Web service, supply the same credentials you use to log on to FlowForce Server.

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:

```plaintext
file:///C:/Program%20Files%20(x86)/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:\FlowForce\ValidateLog.log file. In the event that the log file was not generated, check the log of the job in FlowForce Server to identify the error (see [Viewing the Job Log](#)). 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, in which case the log file cannot be generated.
15.14 Use RaptorXML to Pass Key/Value Parameter Pairs

If RaptorXML Server (or RaptorXML+XBRL Server) runs under FlowForce Server management, you can pass key-value pairs of parameters to execution steps that run XSLT stylesheets. For example, let's assume that you want to run an XSLT stylesheet called transformation.xslt which takes as input a file called books.xml, and two required parameters, "param1" and "param2".

You can create a FlowForce Server jobs to run such an XSLT transformation as follows:

1. In the job configuration page, browse for the xslt function located in the /RaptorXML container. You can also find this function in any container that corresponds to a specific RaptorXML release (for example, /RaptorXML/2020/xslt).

2. Configure the job as shown below.
Note the following:

- In the example above, only some of the parameters are supplied. For a full list of parameters you can supply to the execution step, refer to the RaptorXML Server documentation, see https://www.altova.com/documentation.

- The "Working directory" parameter is set to a directory from where all the input files are read (this makes it possible to enter just the file names instead of full paths). The output file will also be generated in the working directory.

- The XSLT parameters are supplied to the job as key-value pairs. Notice the parameter name and value are entered in separate boxes. Also, the parameter value is enclosed within quotes.

There is also another way to supply parameters, by entering a FlowForce Server expression, for example:

```plaintext
raptopxmlcmdlxmlt_out_name
raptopxmlcmdlxmlt_parameters_name
raptopxmlcmdlxmlt_streaming_serialization_enabled_name
```

```plaintext
library.xml
list(["param1":"1900"],["param2":"Science Fiction"])
```
To use this second approach, the layout of the parameter `raptorxmlcmdl.xslt_parameters_name` must be changed, as follows:

- Click `Set to`, and then click `<Expression>`.

To switch back to the key-value pairs layout:

- Click `Set to`, and then click `<Value>`.
15.15 Generate Multiple PDFs from Multiple XMLs

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

- FlowForce Server is running at the configured network address and port (see Defining the Network Settings)
- 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 mapping job created in this example generates multiple PDF files. 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:\FlowForceWorkingDir).

Demo files used

This example makes use of the following sample files, available at the following path: <Documents>\Altova\MapForce2020\MapForceExamples.

- MultipleInputToMultipleOutputFiles.mfd (the MapForce mapping file)
- PersonListWithGrouping.sps (the StyleVision transformation file)
- Nanonull-Branch.xml, Nanonull-HQ.xml (the source 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\MapForce2020\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 filename 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 <dynamics:“> node means that a new file will be created for every value in the source. The **remove-folder** function is meant to get only the filename (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).

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 appropriate log message if the license is not found or valid.

To deploy the StyleVision stylesheet to FlowForce Server:

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 for the root FlowForce account in
this case; 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.

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.
After the files were successfully deployed, the corresponding entries will appear in the specified FlowForce container (in this case, "/public") when you log on to FlowForce Server:

![Deployment Screen](image)

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 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. 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. The second parameter defines the 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. In this example, the working directory is "C:\FlowForceWorkingDir".

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 The FlowForce Expression Language.

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 the step which executes the mapping and the step which runs the StyleVision transformation. In some cases, it may be necessary to specify separate working directories, in order to avoid file name collisions, and potential job execution errors.

**Running the job**

This job was created as a Web service, so you can run it on demand by clicking a link from the browser. To do this, enter the following URL in the browser address bar: **http://localhost:4646/service/**. Adapt the URL...
accordingly if you configured a different host and port for FlowForce Server, see Defining the Network Settings.

Click the Web service link to execute the job (in this case, "GenerateMultiplePDFs"). When the job finishes, the PDF files generated by the job will be available in the working directory `C:\FlowForceWorkingDir`. Note that, for simplicity, this job did not use error handling. The browser will display "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 see which step has failed and why, see Viewing the Job Log.
16 Glossary

The glossary section includes the list of terms pertaining to FlowForce Server.
16.1 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.

Cluster
A cluster represents several service instances of FlowForce Server that communicate for the purpose of executing jobs in parallel or redistributing jobs if any instance is not available. A cluster consists of one "master" FlowForce Server and one or several "workers".

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, or OAuth security details.
16.2  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.
16.3  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.
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.
16.5 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 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.

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. Job instances are distributed within the cluster as defined by the execution queue associated with the job. A job instance will always run in its entirety on a single cluster member.
Master

A "master" is a FlowForce Server instance that continuously evaluates job-triggering conditions and provides the FlowForce service interface. A master is aware of worker machines in the same cluster and may be configured to assign job instances to them, in addition to (or instead of) processing job instances itself.
16.7  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.
Queue
An execution queue is a "processor" of jobs; it controls how job instances run. In order to run, every job is
assigned to a target execution queue. You can assign a job to an execution queue while configuring the job,
and it will be submitted to that execution queue at runtime. The queue controls how many job instances (of all
the jobs assigned to the queue) can be running at any one time, the delay between runs, and other settings.
Queues can be local to the job, or shared by multiple jobs. When multiple jobs are assigned to the same
execution queue, they will share that queue for executing.
16.9 R

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.
Trigger
When creating a job, you must specify the conditions (or criteria) that will trigger the job, known as triggers. FlowForce Server monitors any defined triggers and executes the job whenever the trigger condition is met.
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.
Web administration interface
The FlowForce Server Web administration interface is the front end of FlowForce Server from where you administer the server and configure jobs. You can access the Web administration interface from a Web browser at the configured address and port.

Worker
A FlowForce Server instance that is configured to communicate with a master instance instead of executing any local jobs. A worker can execute only jobs that a master FlowForce Server has assigned to it.

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.
Index

A

Active Directory,
  integration with FlowForce Server, 79
ADO,
  database connections, 201
ADO.NET,
  database connections, 201
Altova LicenseServer,
  migrating to a new machine, 44
AS2,
  certificate configuration, 238
  concepts, 237
  decryption, 237
  encryption, 226, 237
  encryption settings, 242
  integration with Altova products, 231
  limitations, 226
  mapping from other formats, 231
  mapping to other formats, 231
  message exchange, 259
  overview, 226
  partner configuration, 242
  receiving data, 254
  sending data, 249
  signing, 226, 237
  signing settings, 242
  with FlowForce Server, 226
AS2 service,
  configuring for public access, 254
  creating, 254
  processing requests, 254
  setting permissions, 254
Authentication,
  HTTP, 27, 164
  Windows domain, 27, 164

C

Cache consumer,

Command line interface,
  getting help on, 280
Consumer,
  definition, 460
Container,
  definition, 460
  restricting access to /public, 126
Containers,
  creating, 121
  moving, 121
  overview of, 119
  renaming, 121
  setting permissions on, 124
  viewing permissions of, 123
Credentials,
  adding, 132
  definition, 461

D

Data types,
  in FlowForce, 196
Default time zone,
  configuration of, 77
Digital certificates,
  managing on Windows, 69
  trusting on Linux, 66
  trusting on Mac, 66
  trusting on Windows, 66
Directory polling job,
  example of, 407

E

Encryption,
  within AS2 process, 237
Environment variables,
  setting, 87
Error handling,
Error handling, adding to a job, 412
Error logging, adding to a job, 440
Error/success handling step, definition, 462
Execution result, definition, 462
Execution Step, definition, 470
Expressions, 333

File System Triggers, overview of, 160
FlowForce Server, application data folder, 49 architecture, 21 basic job concepts, 17 basic security concepts, 19 built-in functions, 298 changing the language of, 293 command line interface, 278 configuration of, 48 glossary, 459 localization of, 86, 285 log on to, 27 migrating to a new machine, 44 migrating to latest version of, 290 new features in, 14 opening the setup page of, 51 setting the network address and port of, 53, 55 system requirements, 29 Web administration interface, 24
FlowForce Server Function, definition, 463
FlowForce Web Server, setting the network address and port of, 53, 55
For-each step, definition, 463
FTP, credentials, 131, 134
Function, definition, 463

G
Global Resources, using in FlowForce Server, 213

H
HTTP Triggers, overview of, 162
HTTS connections, configuring FlowForce to accept, 53, 55

I
Input Parameters, adding, 138 built-in, 138 definition, 464 removing, 138
Installation, installation folder (Linux), 34 installation folder (Mac), 37 installation folder (Windows), 43 system requirements, 29 viewing currently installed products (Linux), 34
item, as FlowForce data type, 196

J
Java, configuration, 201
JDBC, database connections, 201
Job, definition, 465
Jobs, caching results of, 167, 429 copying, 137 creating, 136
**Jobs**,  
creating from MapForce mapping, 399  
creating from StyleVision transformations, 432  
credentials, 131, 134  
declaring the return type of, 153  
defining queue settings of, 170  
duplicating, 137  
exporting to another FlowForce Server instance, 171  
exporting to file, 171  
importing from file, 171  
monitoring execution, 180, 181  
status reference, 182  
stopping, 185  
viewing the log of, 187

**K**

Key/value pairs,  
passing to XSLT sheets, 445

**L**

LDAP,  
integration with FlowForce Server, 79  

Linux,  
starting services on, 83  
stopping services on, 83  
trusting server certificates on, 66

**M**

Mac,  
starting services on, 84  
stopping services on, 84  
trusting server certificates on, 66  

Mail parameters,  
configuration of, 78

MapForce Server,  
calling from a job, 448  
inclusion in XSLT, 200, 207  
migrating to a new machine, 44

**R**

RaptorXML functions,  
creating jobs with, 437, 440
Index

RaptorXML Server,
  definition, 469
  integration with, 223
RaptorXML(XBRL) Server,
  migrating to a new machine, 44
RaptorXML+XBRL Server,
  definition, 469
Resources,
  creating, 213
  editing, 213
result,
  as FlowForce data type, 196
results,
  as FlowForce data type, 196
Role,
  definition, 469
Roles,
  assigning roles to, 108
  assigning to users, 107
  built-in roles, 106
  creating, 104
  renaming, 106
root user,
  resetting the password of, 108

T

Timer Triggers,
  overview of, 159
Trigger,
  Definition, 471
triggerfile parameter,
  purpose of, 138
Triggers,
  creating, 158
  deleting, 158

U

UNC,
  syntax in paths, 157, 298
Users,
  built-in roles, 106
  creating, 103
  renaming, 106

W

Web administration interface,
  definition, 473
Web Services,
  authentication, 164
  exposing jobs as, 164, 417
Windows,
  referring to network paths on, 298
  starting services on, 85
  stopping services on, 85
  trusting server certificates on, 66
Windows domain users,
  importing into FlowForce Server, 104
Working directory,
  definition, 473
  usage, 207

@ 2019 Altova GmbH