Automating Data Integration Workflows

with Altova FlowForce Server

ALTOVA®

Automating Data Integration Workflows with Altova FlowForce Server

David McGahey

Altova, Inc.

www.altova.com

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2014

© 2014 Altova GmbH

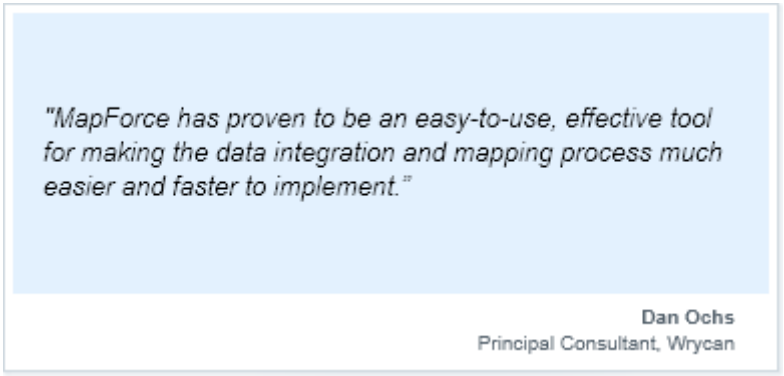
ISBN 978-1-933210-90-2

Table of Contents

Introduction

About this e-book

Links for even more info on FlowForce Server



This e-book collects posts from the Altova blog and information originally published on the Altova Web site to provide an overview of the functionality of the Altova FlowForce Server platform.

FlowForce Server is a highly-customizable platform to automate data transformations defined by MapForce data mappings, report and document generation, and other tasks on dedicated servers, virtual machines, or workstations scaled for the scope of the project. FlowForce Server empowers data architects, analysts, and other IT professionals to efficiently complete enterprise-level data integration tasks.

There is a lot more information about FlowForce Server on the Altova Web site. Check out the:

- FlowForce Server product feature pages
- MapForce graphical data mapping tool

And if you're not a FlowForce Server user already, visit the Altova Download Center for a free, fully-functional trial.

Chapter 1

Introducing Altova FlowForce Server

An exciting new platform for enterprise data transformation, reporting, and other tasks

Altova FlowForce Server is an exciting new platform for execution of automated data mappings that is designed to provide comprehensive management and control over data transformations performed by dedicated high-speed servers, virtual machines, or even regular workstations, depending on the size of the task.

While royalty-free code generation and the MapForce API and StyleVision API can assist with automation of repeated transformations, FlowForce Server provides much greater power and flexibility. FlowForce Server is a server-based tool with a Web interface that makes it much easier to implement, manage, or modify data transformation jobs in a busy data processing environment.

FlowForce Server can administer multiple transformation jobs simultaneously, lets users define and adjust a variety of job triggers and actions on the fly, can perform housekeeping tasks like moving output files or cleaning up intermediate work, records detailed logs of all activity, and much more.

FlowForce Server features robust access control for jobs and related data files, so departments can work independently without seeing or overwriting each other's data. Access control functionality includes defined Users and Roles, Privileges, and Credentials, all managed by FlowForce Server Administrators.

FlowForce Server even supports remote job requests via an HTTP client and job parameters that can be passed to any step in a job. When used together with the request interface, job parameters empower users to specify input values in the job request.

FlowForce Server consists of a set of components that work together as illustrated in the diagram below.

The **FlowForce Server** continuously checks for trigger conditions, starts and monitors job execution, and writes detailed logs. The **FlowForce Web Administration Interface** application runs in an internet browser and provides the front-end for communication with FlowForce Server. To take full advantage of server resources and meet the demands of busy data transformation workflows, multiple jobs – even multiple instances of the same job – can run simultaneously on FlowForce Server.

MapForce Server performs data transformations based on preprocessed and optimized data mappings stored in MapForce Server Execution files prepared by MapForce and uploaded over a network. Preprocessing enables faster performance and reduced memory footprint for most data mappings.

StyleVision Server is based on the built-in report and document generation engine developed for StyleVision. StyleVision Server renders XML and/or XBRL data into HTML, RTF, PDF, or Microsoft Word files based on StyleVision stylesheets and supporting design elements.

RaptorXML Server is the third-generation, hyper-fast XML and XBRL processor from the makers of XMLSpy. RaptorXML provides strict conformance with all relevant XML and XBRL standards, including support for the very latest recommendations, and has been submitted to rigorous regression and conformance testing. To meet industry demands for an ultra-fast processor that can handle the huge amounts of XML and XBRL data being generated, RaptorXML takes advantage of the processing power afforded by the multi-CPU, multi-core computers and servers.

Cross-platform compatibility All server components described above are available for Windows, Linux, and Mac OS platforms.

FlowForce Server empowers data architects, analysts, and other IT professionals to efficiently accomplish today’s complex enterprise-level data integration tasks.

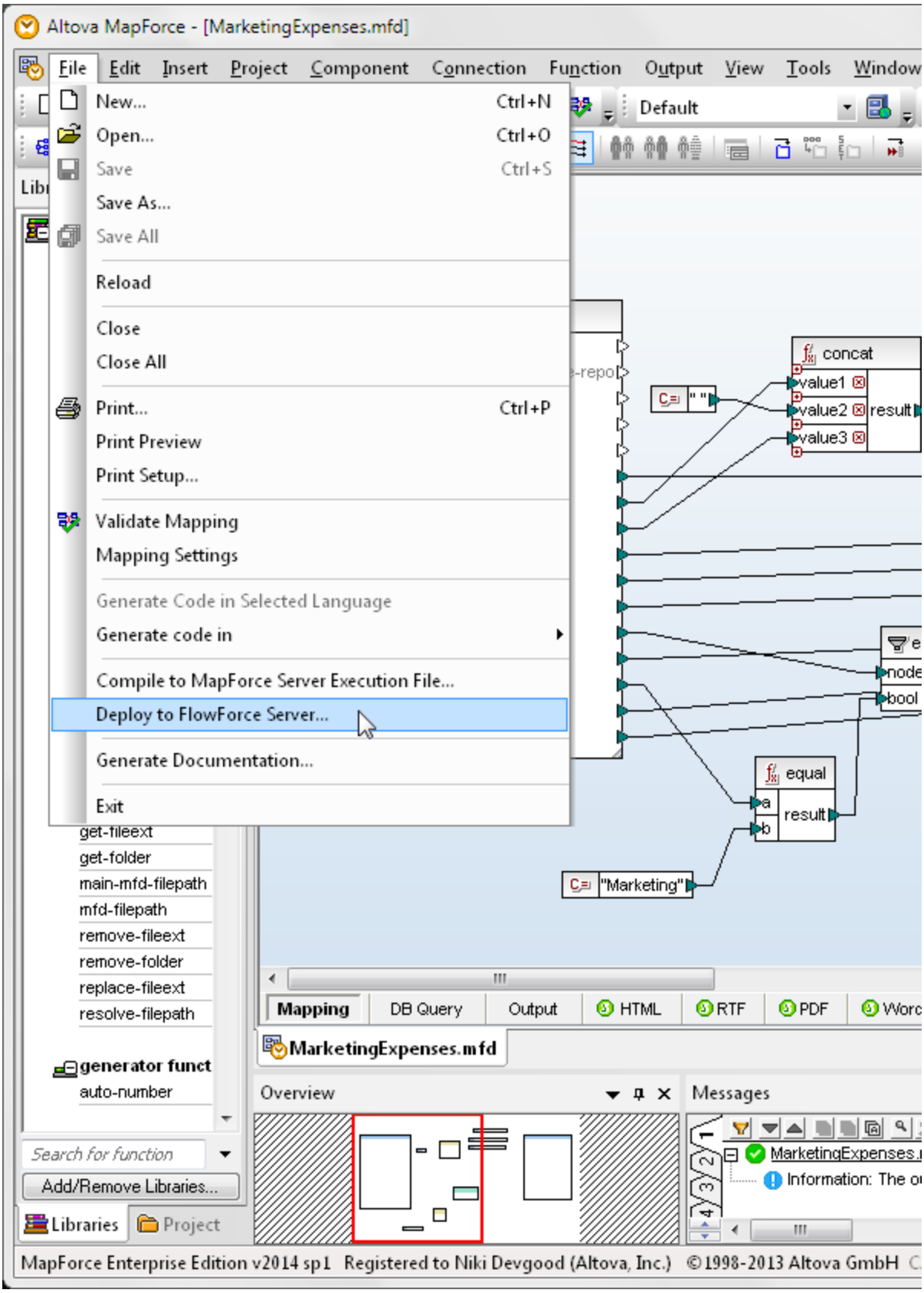
Chapter 2

Deploy Data Mappings and Report Designs for Automated Processing

Deploy directly from MapForce and StyleVision

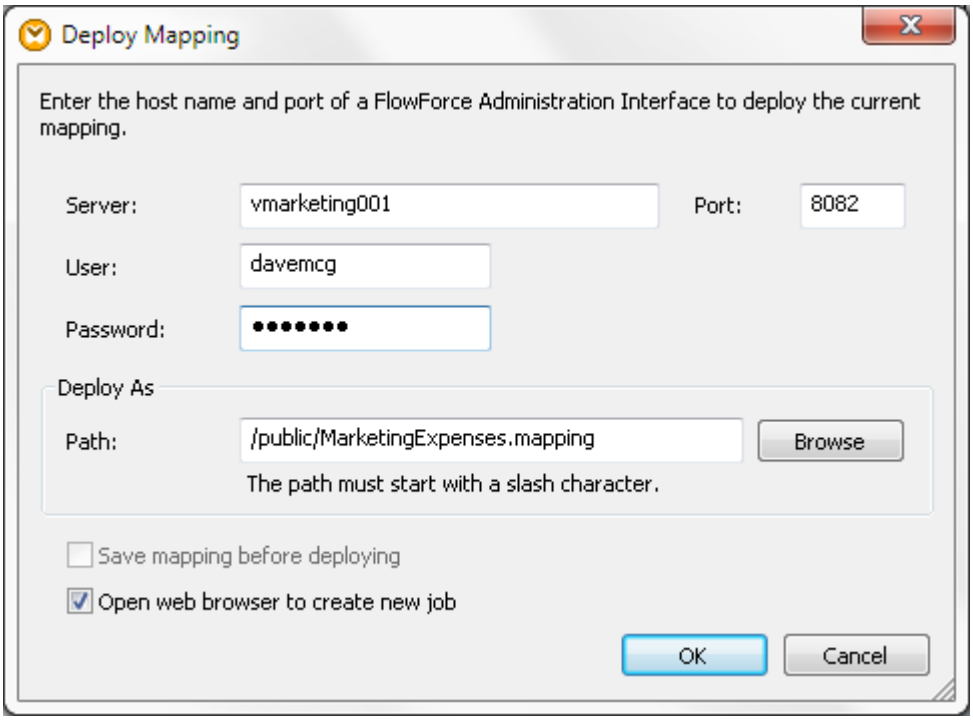
Deploying data mappings created in MapForce and report designs created in StyleVision for automated processing by FlowForce Server is straightforward and quick.

The File menu in MapForce includes two options to optimize, preprocess, and deploy data mappings for MapForce Server and FlowForce Server. Preprocessing enables faster performance and reduced memory footprint for most data mappings.

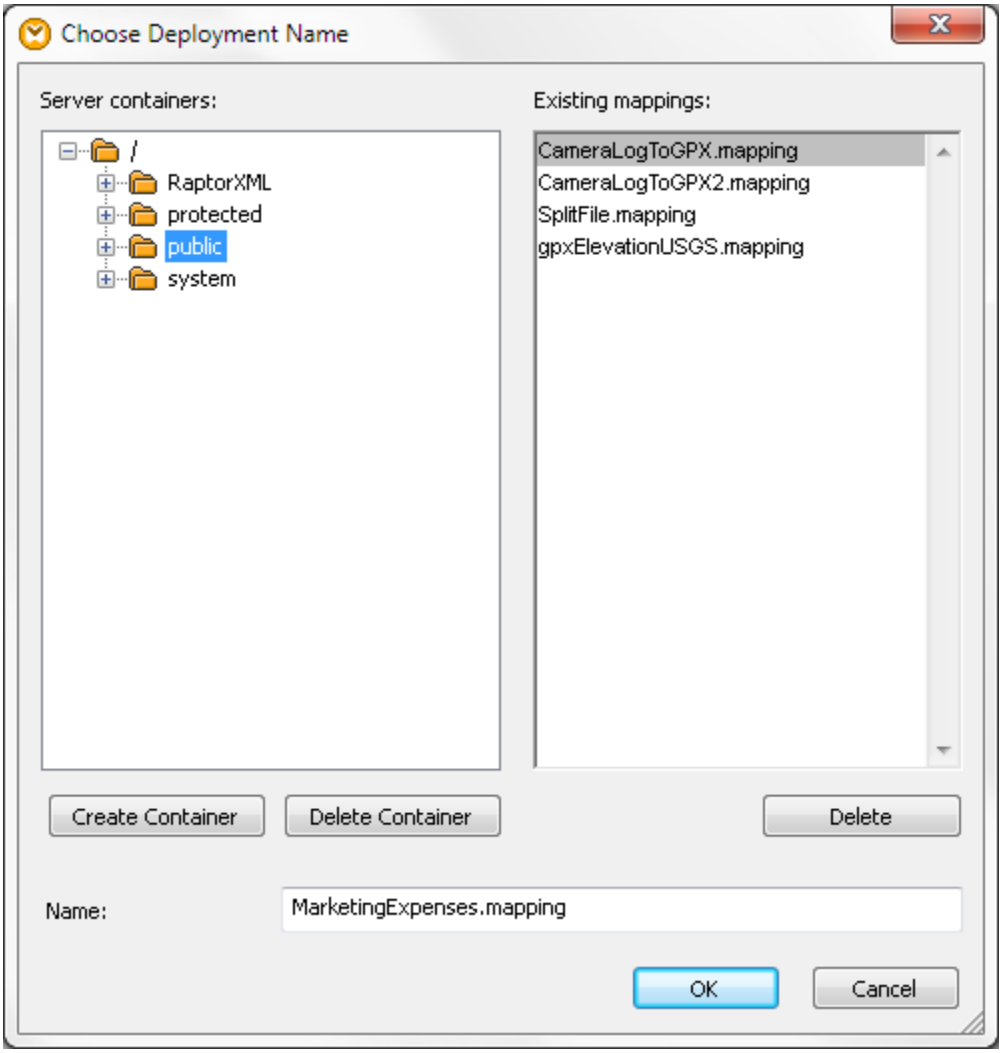

Altova MapForce - [MarketingExpenses.mfd]
File
Edit
Insert
Project
Component
Connection
Function
Output
View
Tools
Window
New... Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As...
Save All
Reload
Close
Close All
Print... Ctrl+P
Print Preview
Print Setup...
Validate Mapping
Mapping Settings
Generate Code in Selected Language
Generate code in
Compile to MapForce Server Execution File...
Deploy to FlowForce Server...
Generate Documentation...
Exit
Default
concat
value1
value2
value3
result
" "
equal
a
b
result
"Marketing"
node
bool
get-fileext
get-folder
main-mfd-filepath
mfd-filepath
remove-fileext
remove-folder
replace-fileext
resolve-filepath
generator funct
auto-number
Search for function
Add/Remove Libraries...
Libraries
Project
Mapping
DB Query
Output
HTML
RTF
PDF
MarketingExpenses.mfd
Overview
Messages
MapForce Enterprise Edition v2014 sp1 Registered to Niki Devgood (Altova, Inc.) ©1998-2013 Altova GmbH

Compile to MapForce Server Execution File saves a local file for MapForce Server running in a standalone configuration executed from a command line. Creating the execution file is nearly instantaneous.

Deploy to FlowForce Server opens a dialog that allows users to connect directly to FlowForce Server and log in to create and deploy the .mfx, as shown below:

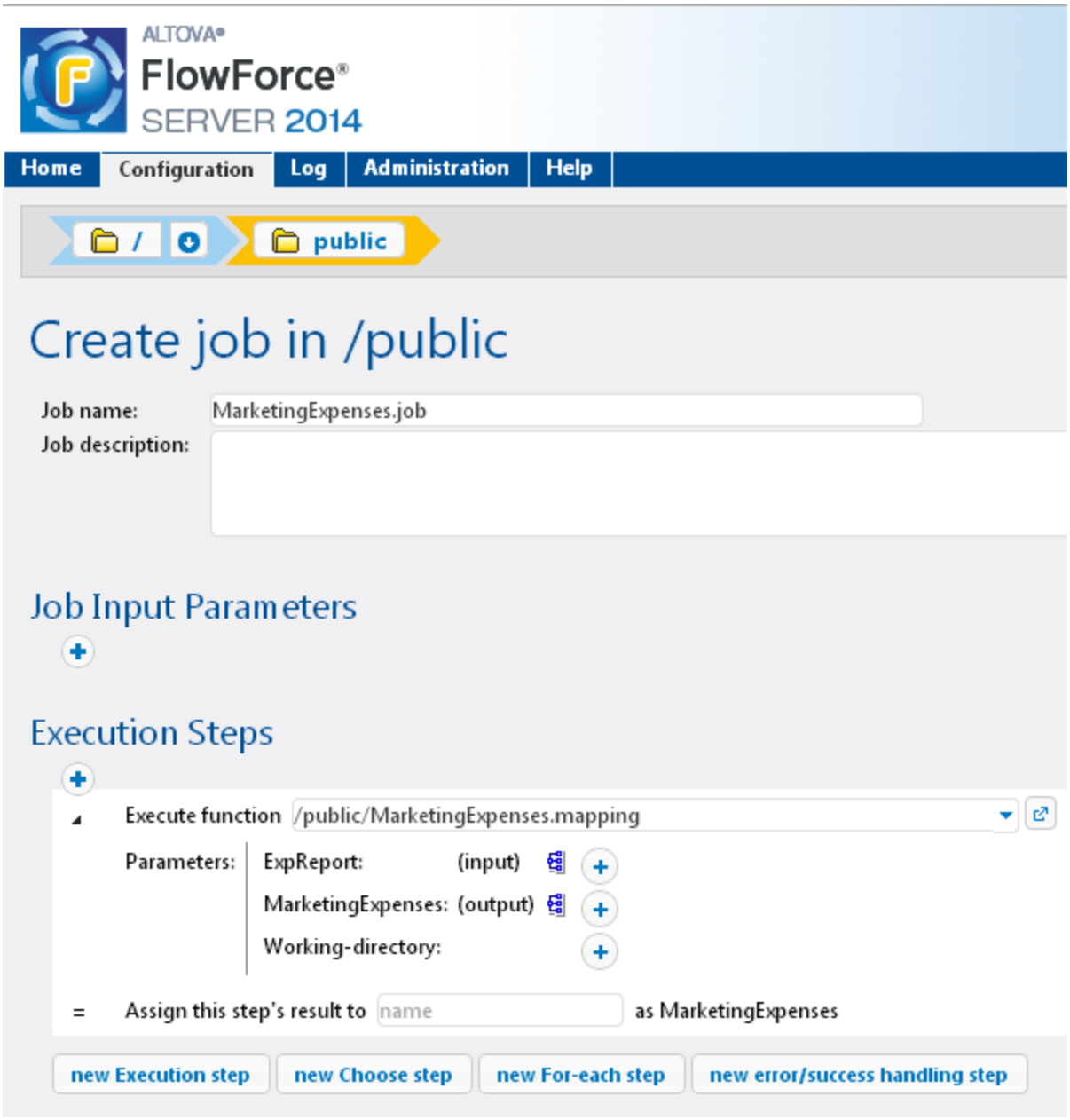


Note options in this dialog to choose the destination directory or rename the mapping. The connection to deploy mappings conforms to all FlowForce Server security functionality, so permissions are managed by FlowForce Server settings. The Browse button lets users examine existing folders and data mappings on the FlowForce Server:



Replace defined input and output file names

The data mapping becomes a FlowForce Server job execution step, with parameters to assign input and output file names for automated processing. This simplifies developing and reusing MapForce data mappings, as users focus on the design and test with local files containing sample data.



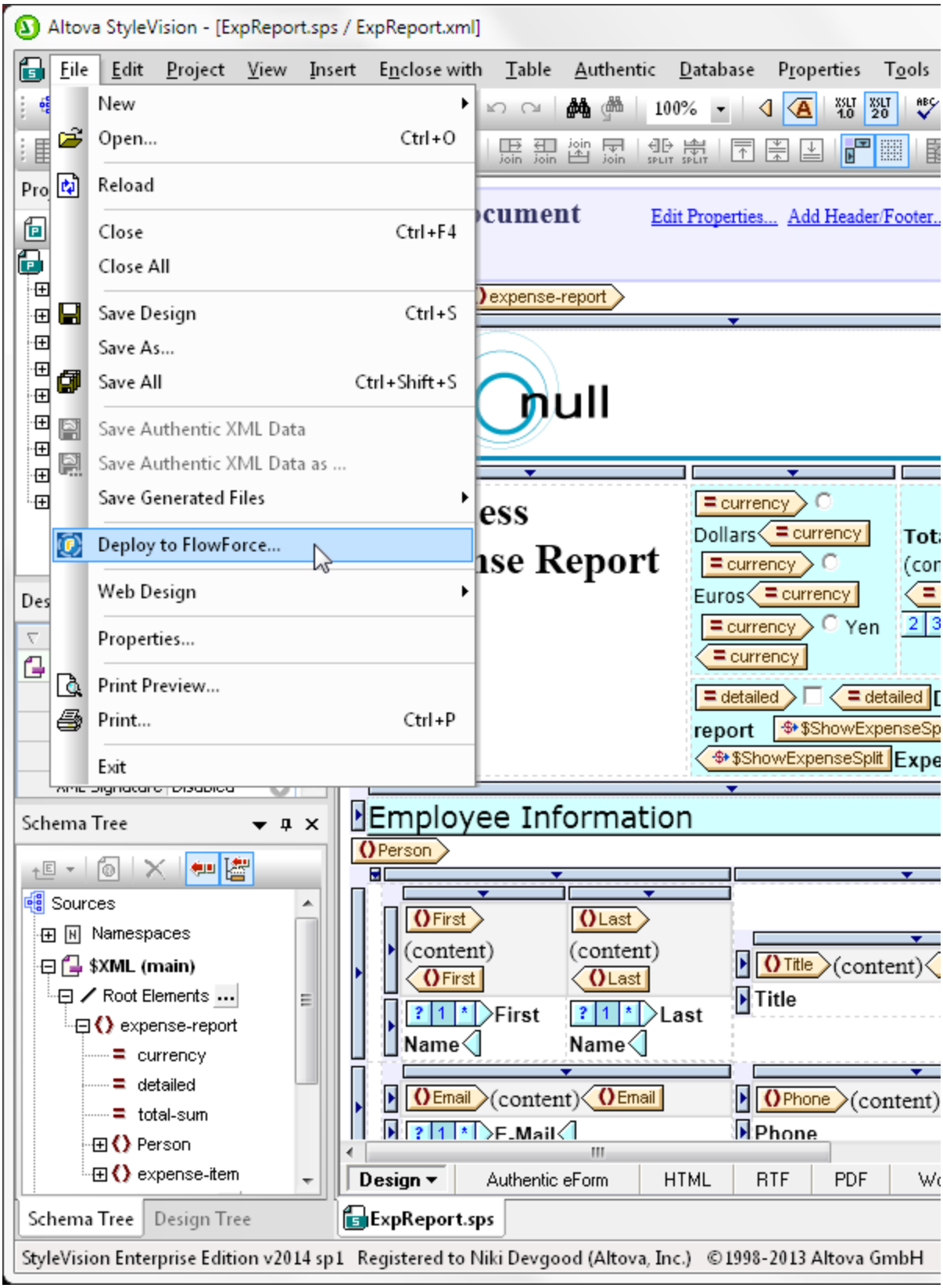
Input and output parameter names in the FlowForce Server job definition correspond to components defined in the original MapForce mapping.

From Stylesheets to PXF

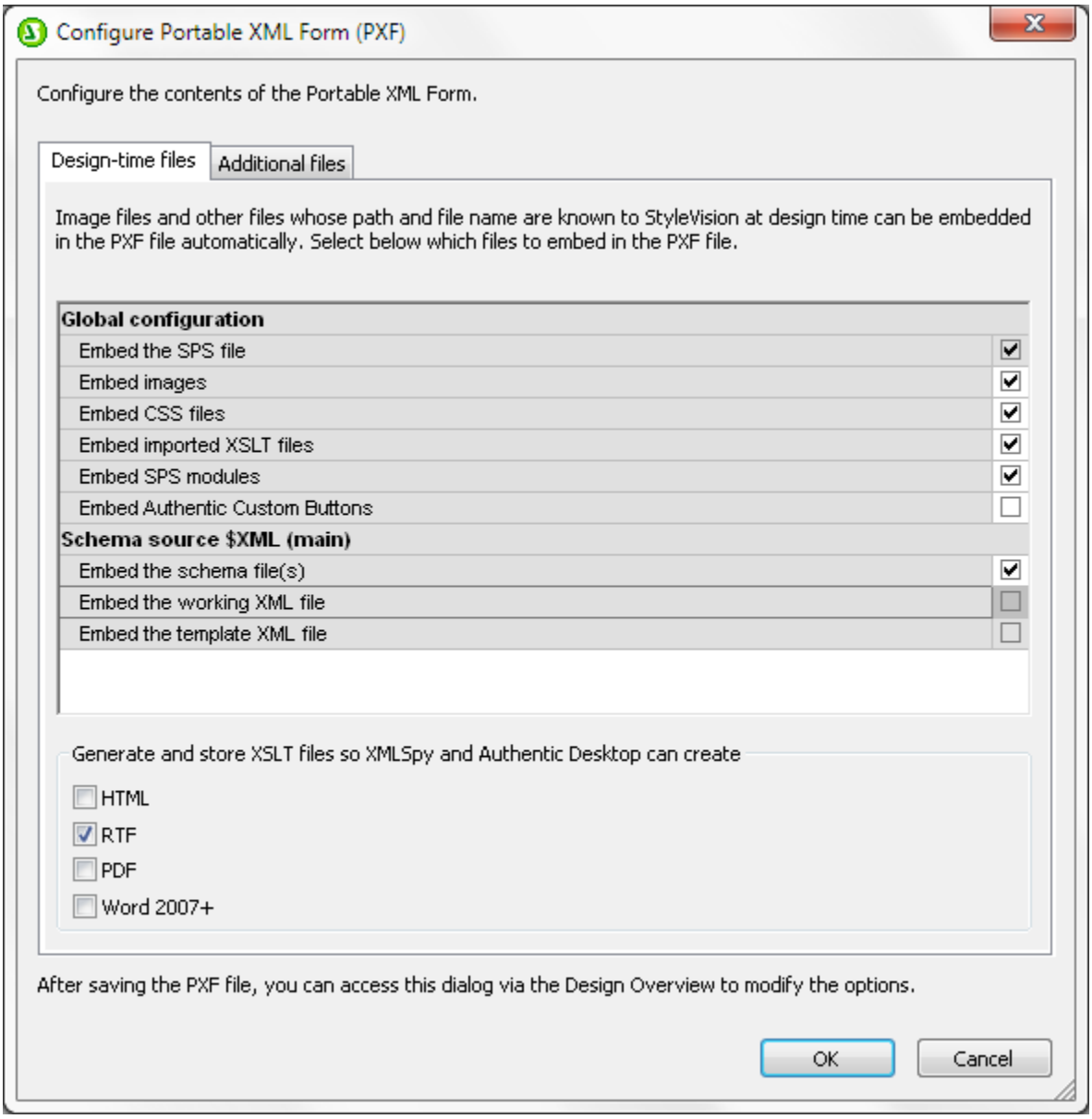
Altova originally introduced the .pxf (Portable XML Form) file format to conveniently package StyleVision SPS stylesheets with all files required by the design, including the XML schema file, source XML file, image files used in the design, and XSLT files for

transformation output formats. FlowForce Server uses .pxf files with StyleVision Server to render output in HTML, RTF, PDF, or MS Word formats.

The File menu in StyleVision includes an option to Deploy to FlowForce Server that first opens a dialog allowing users to customize files contained in the .pxf.


Altova StyleVision - [ExpReport.sps / ExpReport.xml]
File
Edit
Project
View
Insert
Enclose with
Table
Authentic
Database
Properties
Tools
New
Open... Ctrl+O
Reload
Close Ctrl+F4
Close All
Save Design Ctrl+S
Save As...
Save All Ctrl+Shift+S
Save Authentic XML Data
Save Authentic XML Data as ...
Save Generated Files
Deploy to FlowForce...
Web Design
Properties...
Print Preview...
Print... Ctrl+P
Exit
100%
Edit Properties...
Add Header/Footer...
expense-report
null
Dollars
Euros
Yen
currency
detailed
report
$ShowExpenseSplit
Employee Information
Person
First
Last
Name
Title
Email
Phone
Schema Tree
Sources
Namespaces
$XML (main)
Root Elements
expense-report
currency
detailed
total-sum
Person
expense-item
Design
Authentic eForm
HTML
RTF
PDF
Schema Tree
Design Tree
ExpReport.sps
StyleVision Enterprise Edition v2014 sp1 Registered to Niki Devgood (Altova, Inc.) ©1998-2013 Altova GmbH

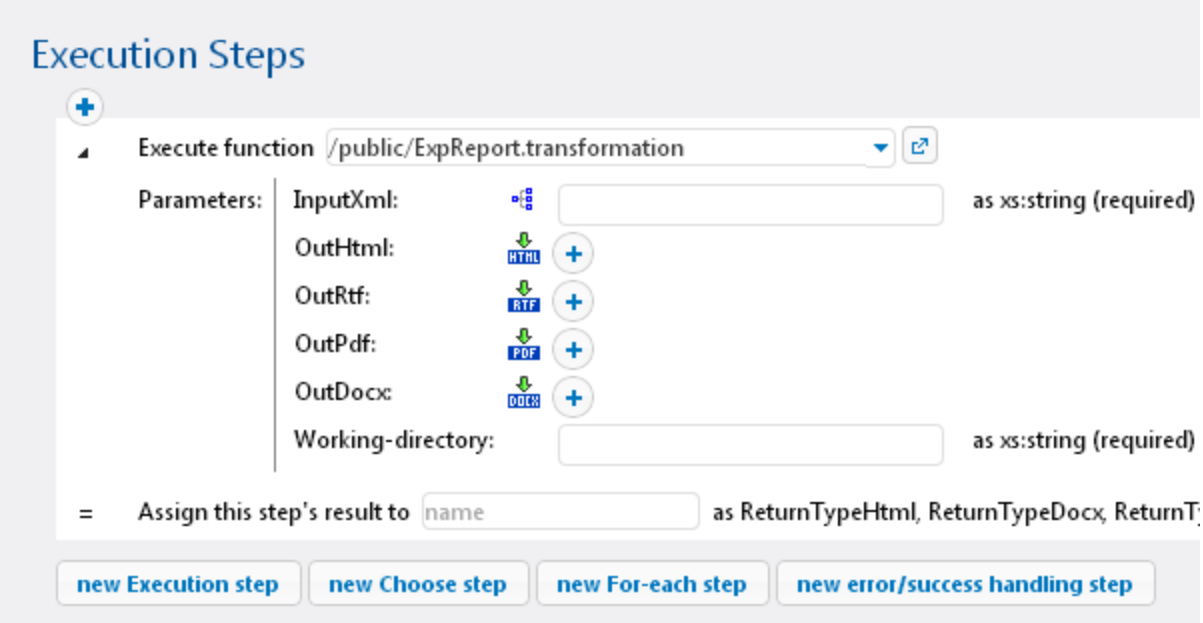
For example, the original working XML source file and XSLT files for unneeded output formats might be omitted.



After the .pxf file is configured, StyleVision connects to FlowForce Server with a dialog similar to the MapForce connection shown above.

Parameters for Transformation File Names

The .pxf file becomes a FlowForce Server transformation job step, where files for input data and each potential output format are assigned as part of the job definition.



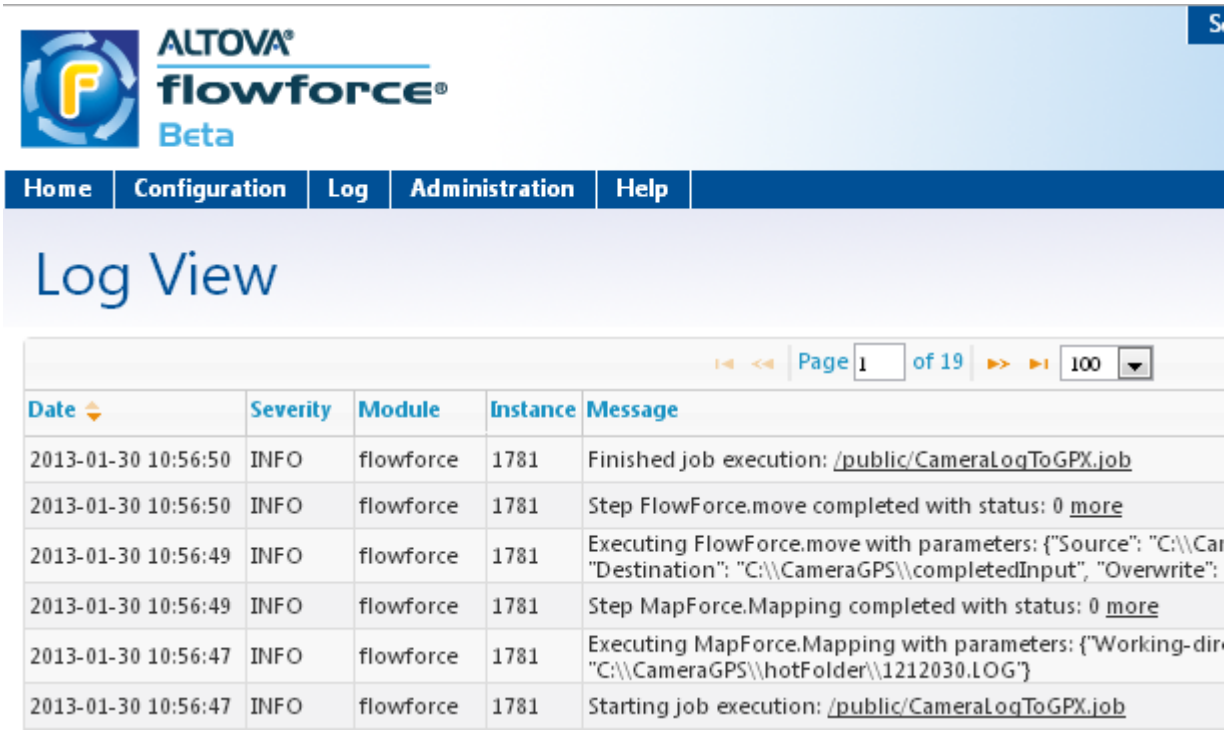
Using parameters for file names empowers the designer working in StyleVision to focus on creating the most compelling and rich document to present the data at hand, while preserving flexibility at the server.

Chapter 3

Automate Data Mappings

One example of an automated data transformation

Altova designed FlowForce Server to provide comprehensive automation, management, and control over data transformations performed by dedicated high-speed servers. FlowForce Server can provide hot folder automation of data mappings and maintains a detailed activity log users can monitor remotely in a Web browser window. The screenshot below shows the log for FlowForce Server running the MapForce data mapping CameraLogToGPX we wrote about in the blog post titled Process Multiple Input Files in a Single Data Mapping. This mapping used wildcards to specify multiple input files for processing.



It only takes a few minutes to set up, run, and review the results of jobs like this on FlowForce Server.

Wildcards or Hot Folders?

Wildcards and hot folders increase the complexity of a data transformation workflow, and using them successfully requires careful planning. Let's take a minute to look a little deeper at the scenario we want to implement.

Assume we are the IT department in a company that publishes nature and hiking guides. We employ photographers who go out trekking and record their routes as they go, using the GPS tracking feature of their digital cameras. We want to convert the camera GPS log files to XML-based .gpx format for mapping and other processing.

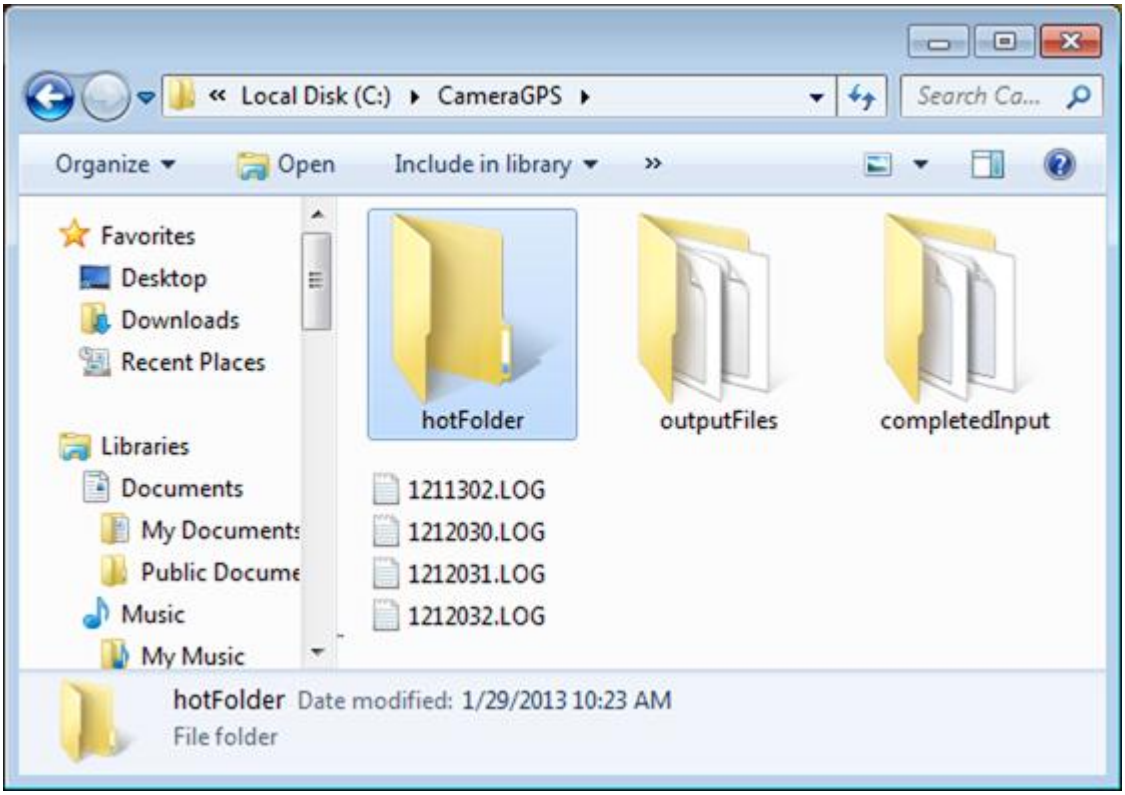
We will publish a folder on our network where photographers can drop off their GPS log files. This will be the hot folder FlowForce Server watches for new files to supply as input to the CameraLogToGPX mapping.

We only need to process each input file once. So, after the data transformation is complete, we can remove the input file from the hot folder. We also want to place the output file in a separate folder.

This suggests the following FlowForce Server job steps:

- Look in the hot folder to see if new a input files has arrived
- Perform the data mapping on the input file and place the output file in a separate folder
- Move the input file to a permanent location

The diagram below shows a folder structure we can use for the workflow, with files ready to drop into the hot folder for processing:



The hot folder is C:\CameraGPS\hotFolder and the generated .gpx files will be placed in C:\CameraGPS\outputFiles. When the data mapping is done, input files will be moved to C:\CameraGPS\completedInput.

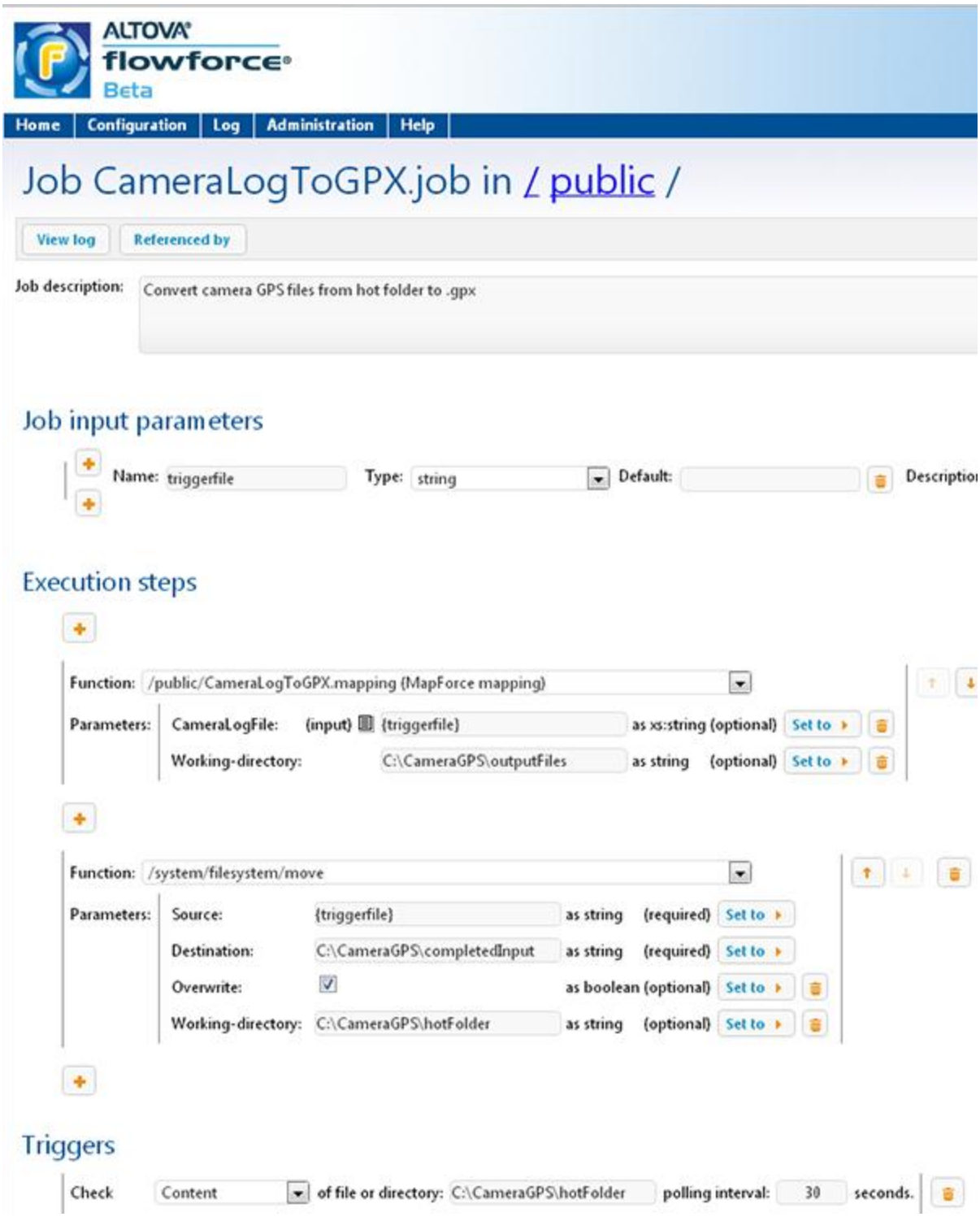
Deploy the Mapping on a Server

We need to plan for the filenames of the input and output files, so we can instruct FlowForce Server to provide the input filename as a job parameter as new files arrive in

the hot folder for processing. We will also want FlowForce to specify the location of the output file.

Defining the Job in FlowForce Server

The screenshot below shows the complete job steps defined in a FlowForce Server job properties window:



The job trigger is defined at the bottom of the window. Every 30 seconds, FlowForce Server will check the hot folder. If the contents have changed, FlowForce Server will

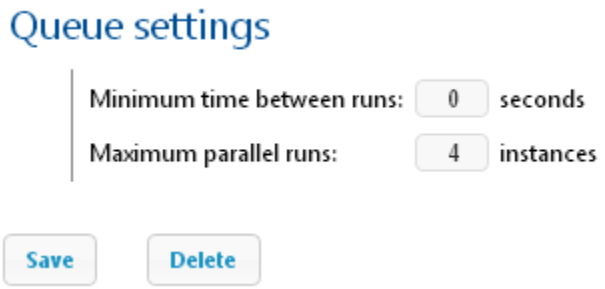
execute the job steps. Each Execution step could be a deployed MapForce mapping, a system step, or even another FlowForce Server job.

The name of each new file entering the hot folder becomes the parameter called {triggerfile} that we use in the mapping step as the input filename, and in the move step as the name of the file to be moved.

The Working-directory parameter in the mapping step defines where the output files will be placed.

FlowForce Server also includes features to set automatic run and stop times for jobs, user permissions and roles, and Queue settings to define minimum time between job runs and maximum parallel instances of a job.

In our scenario, we are likely to receive multiple input files in groups as they are copied from photographers' memory cards. Multiple parallel runs can greatly improve throughput. As a rule of thumb, you might want to match the number of cores or CPUs in the machine running FlowForce Server.

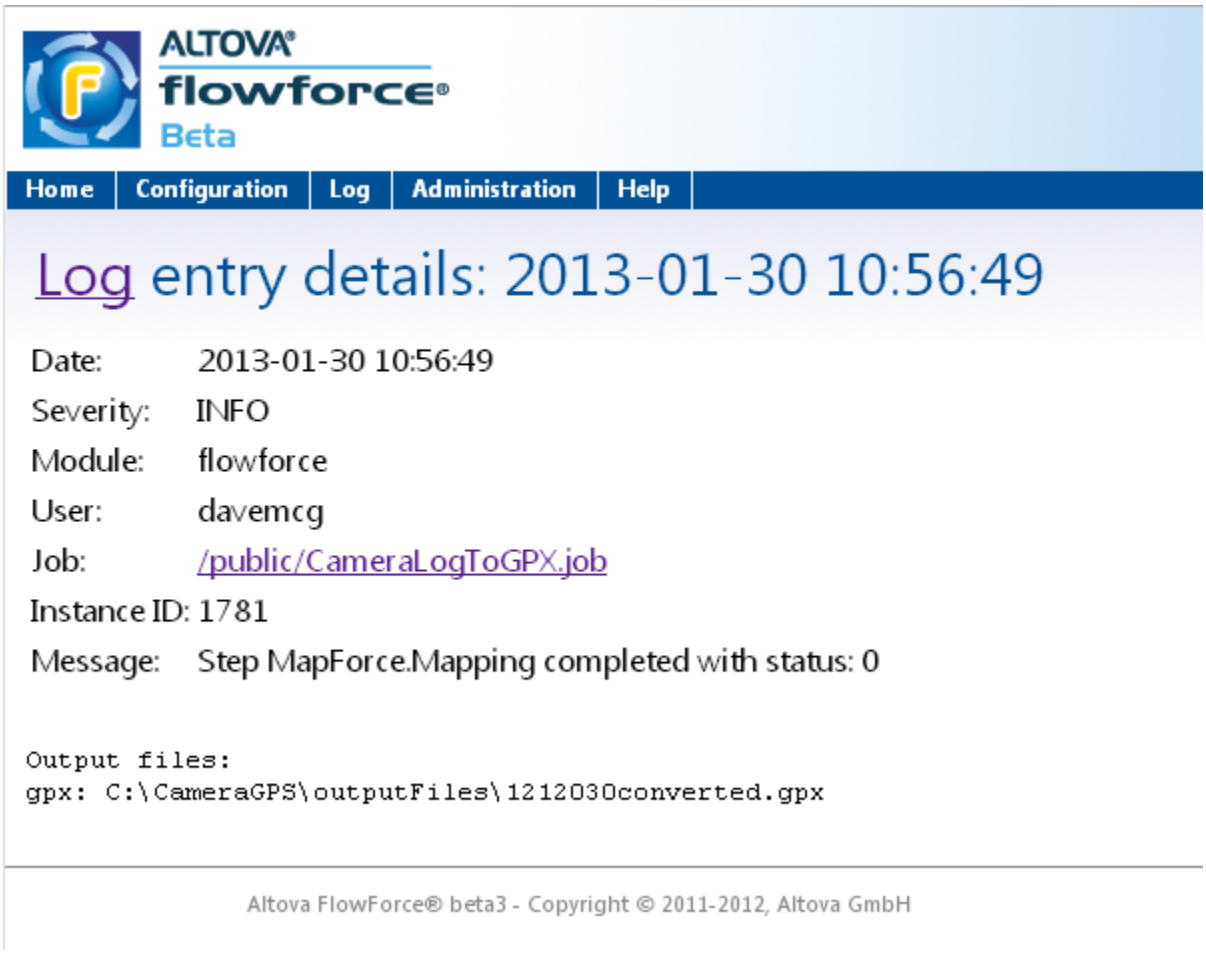


Exploring the Job Log

Each time FlowForce Server runs our job, six lines are added to the Log View shown in the illustration at the top of this post. The first and last lines record the start and completion of the job, and each Execution step generated its own start and completion messages. The phrase "completed with status: 0" means the step was successful with no errors.

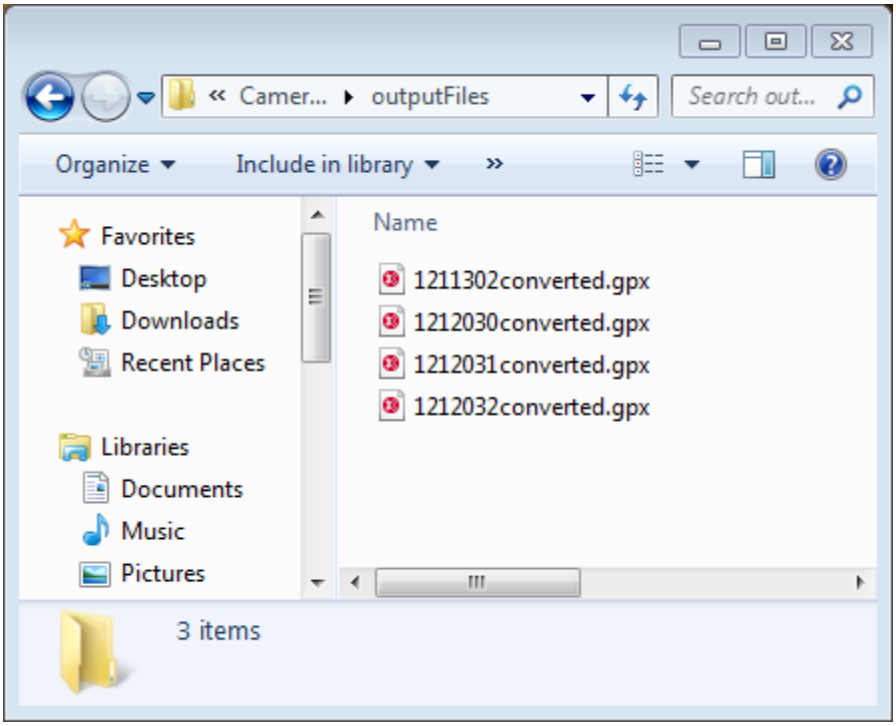
2013-01-30 10:56:49	INFO	flowforce	1781	Step MapForce.Mapping completed with status: 0 more
2013-01-30 10:56:47	INFO	flowforce	1781	Executing MapForce.Mapping with parameters: {"Working-dir "C:\\CameraGPS\\hotFolder\\1212030.LOG"}
2013-01-30 10:56:47	INFO	flowforce	1781	Starting job execution: /public/CameraLogToGPX.job

We can click the more links for a detailed report on each Execution step. The screenshot below shows the message for the MapForce mapping Execution step:



Each file dropped into the hot folder generates an individual FlowForce Server job instance, even if multiple files are added as a group. This makes it easy to track any individual input file that generates an error.

When we dropped four files into the hot folder, FlowForce Server ran four jobs, and the contents of the output folder looked like this:


« Camer... ▸ outputFiles
Search out...
Organize
Include in library
»
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Name
1211302converted.gpx
1212030converted.gpx
1212031converted.gpx
1212032converted.gpx
3 items

Chapter 4

Automate Report Generation

Adding an automated report based on transformed data

A FlowForce Server job can automate a complete data transformation workflow by executing MapForce Server for data mapping and pipelining results to StyleVision Server to render a variety of output formats.

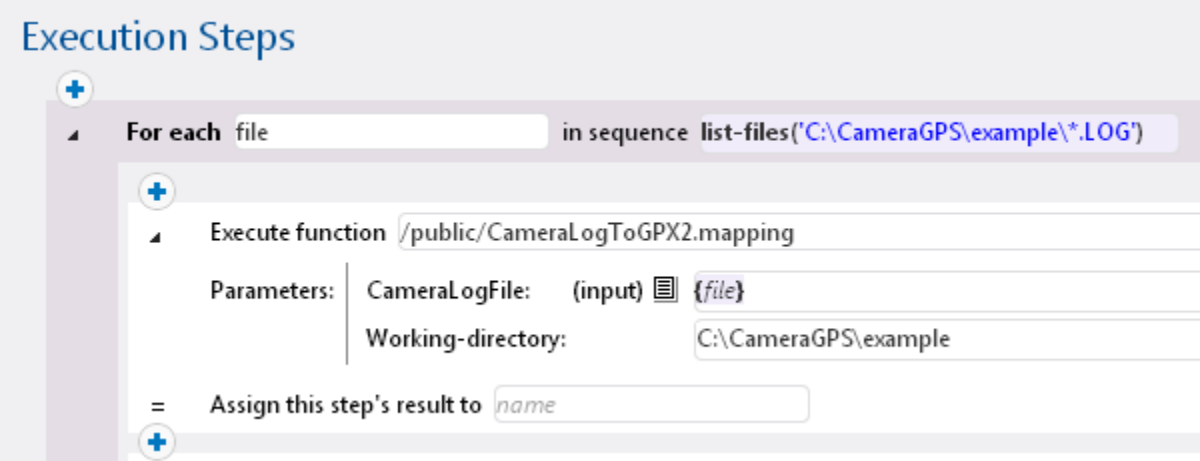
This chapter describes a variation on the data mapping described in the previous chapter. This time we will look at a FlowForce Server job that performs data mapping and transformation to generate a report. The job steps are illustrated in these messages from the FlowForce Server Log, with the most recent step at the top of the list:

Instance	Message
135	Step StyleVision.TransformationID completed with status: 0 more
	Selected tool: StyleVision 2013r2.
135	Executing StyleVision.TransformationID with parameters: {"OutPdf": "", "InputXml": "C:\\CameraGPS\\example\\1211300.LOG.gpx", "OutDocx": "", "Working-directory": "C:\\CameraGPS\\example\\workFiles", "OutRtf": "", "OutHtml": "C:\\CameraGPS\\example\\1211300.LOG.gpx.html"}
135	Step MapForce.Mapping completed with status: 0 more
	Selected tool: MapForce 2013r2.
135	Executing MapForce.Mapping with parameters: {"Working-directory": "C:\\CameraGPS\\example", "CameraLogFile": "C:\\CameraGPS\\example\\1211300.LOG"}

We'll start with the GPS log files created by a digital camera. We will map the log files to GPX format, and we'll use the mapping output with a StyleVision SPS stylesheet adapted from the XPath Enhances XML Reports blog post to produce a time and elevation report for each file.

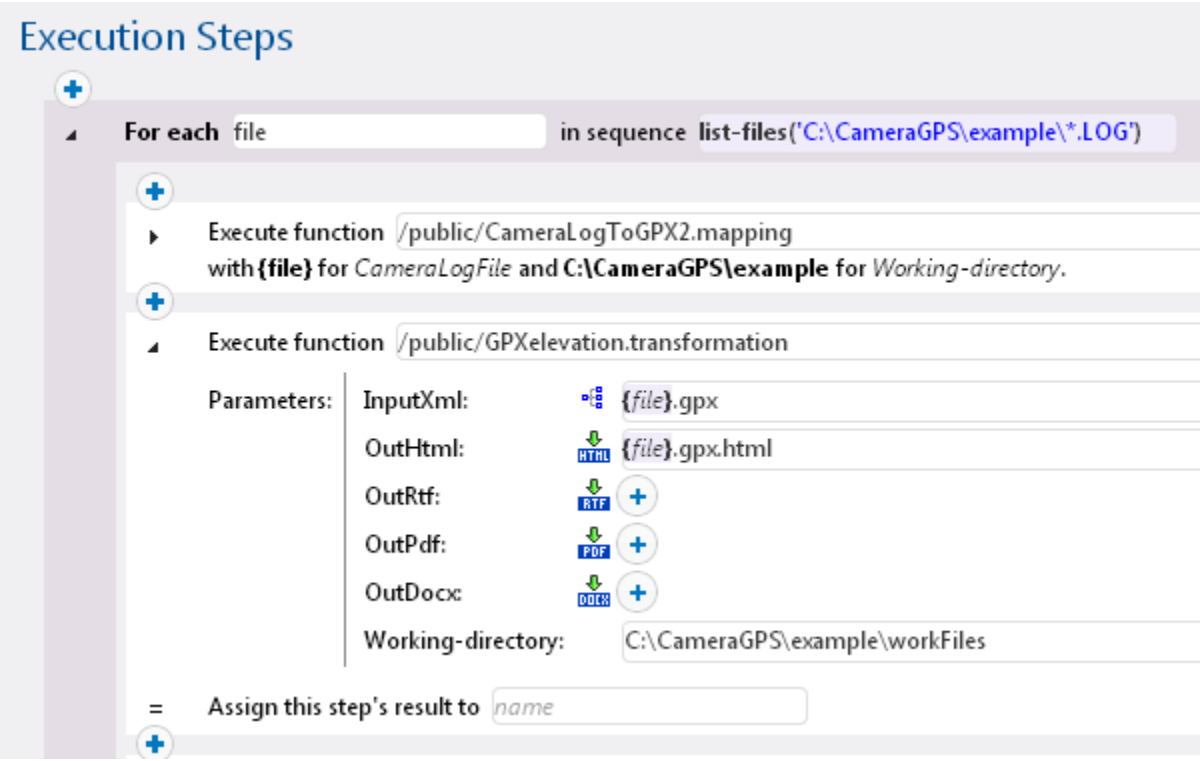
This time, instead of using a hot folder, we will poll the input folder on a regular schedule. A FlowForce Server *For-each* job step repeats based on the result of an

expression. We can use *For-each* to build a list of files in a folder, then repeat one or more steps for each file. Here is how it looks in the job configuration page:



The line labeled Execute function defines the mapping to be used by MapForce Server, and the input parameter {file} refers to each file in the list C:\CameraGPS\example*.LOG.

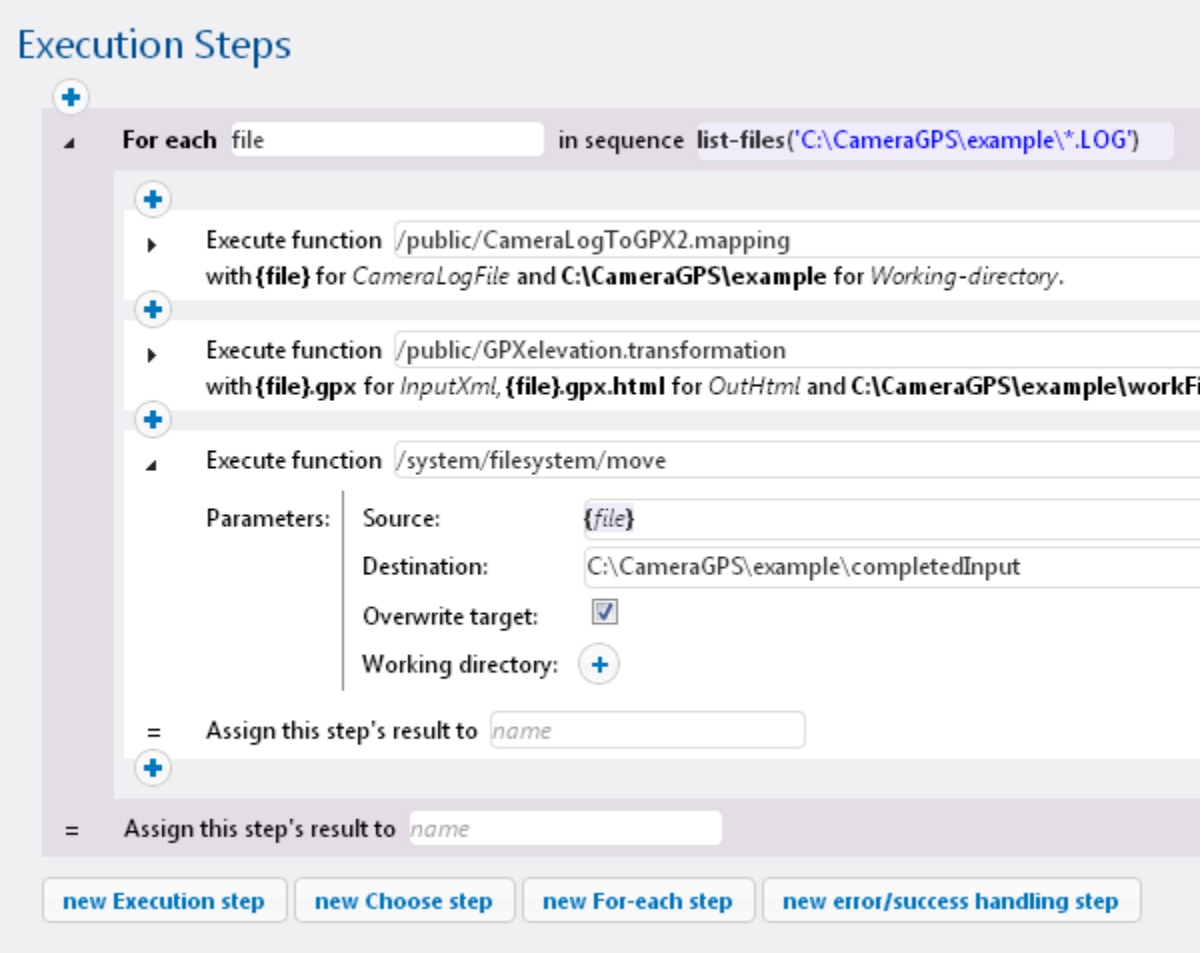
We can add an execution step to instruct StyleVision Server to perform the transformation:



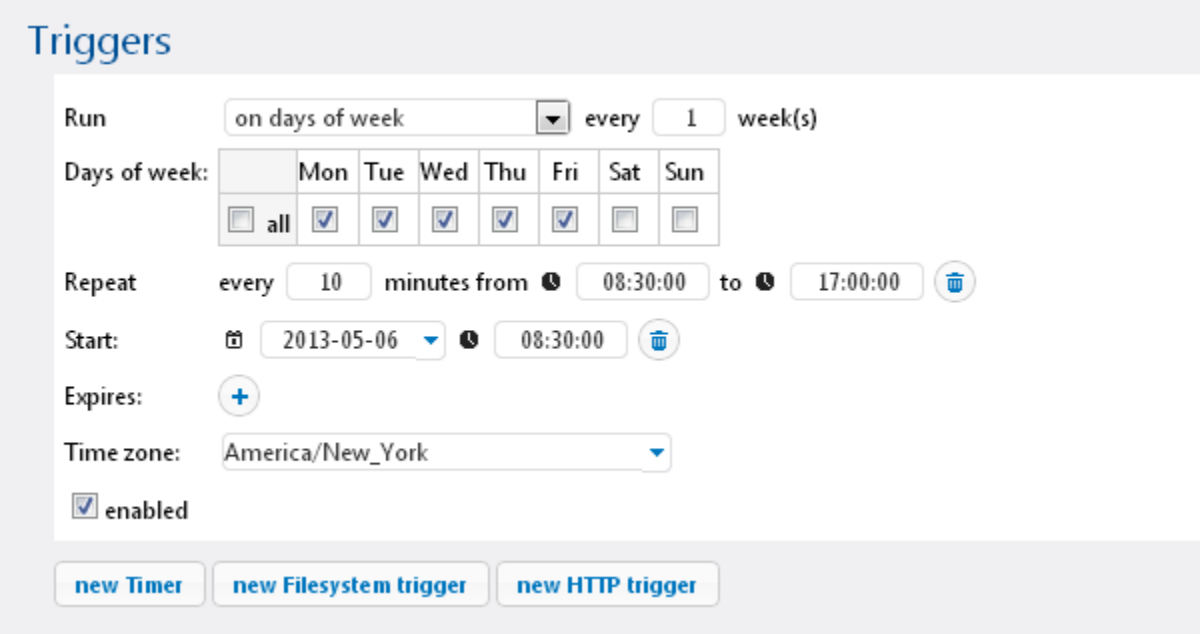
The data mapping creates output files by adding .gpx to the name of the input file, and now we can define the transformation input using the {file} variable with the new file suffix. We chose to create .html output, but we could just as easily create other formats for a multi-channel publishing implementation.

The transformation working directory is the location where StyleVision Server unpacks the contents of the .pxf file containing the stylesheet, XML Schema, and other needed components. Using a dedicated working folder will keep the workflow more organized.

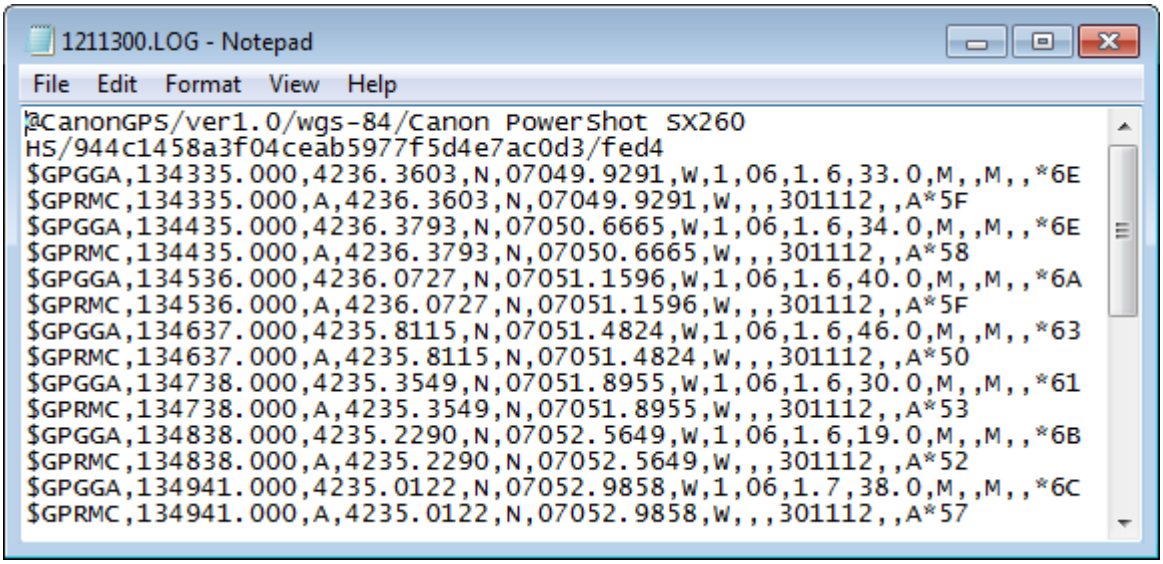
We want to allow network users to drop new .LOG files into the C:\CameraGPS\example folder and we want run the FlowForce Server job on a regular schedule, but we don't want to process the same files over and over. We can define one more job step to move the processed file to a different location:



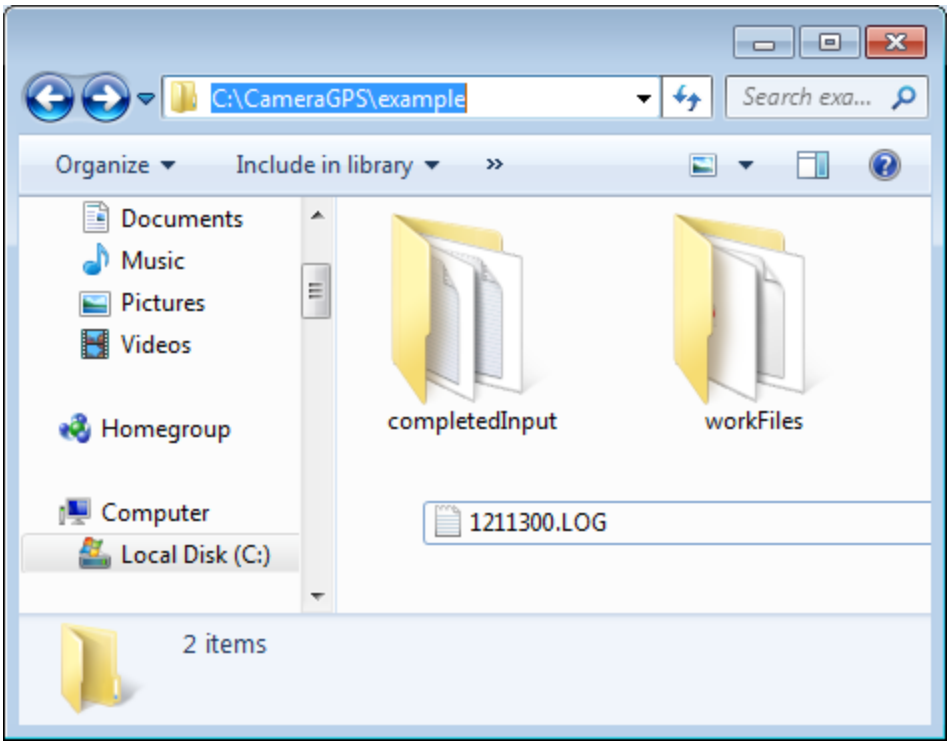
The complete FlowForce Server job is a series of three steps that loops for each .LOG file found in the folder. We can set up a repeating trigger for the workweek or any other appropriate schedule:



Here is a portion of a .LOG file created by the camera that is an example of one input file:



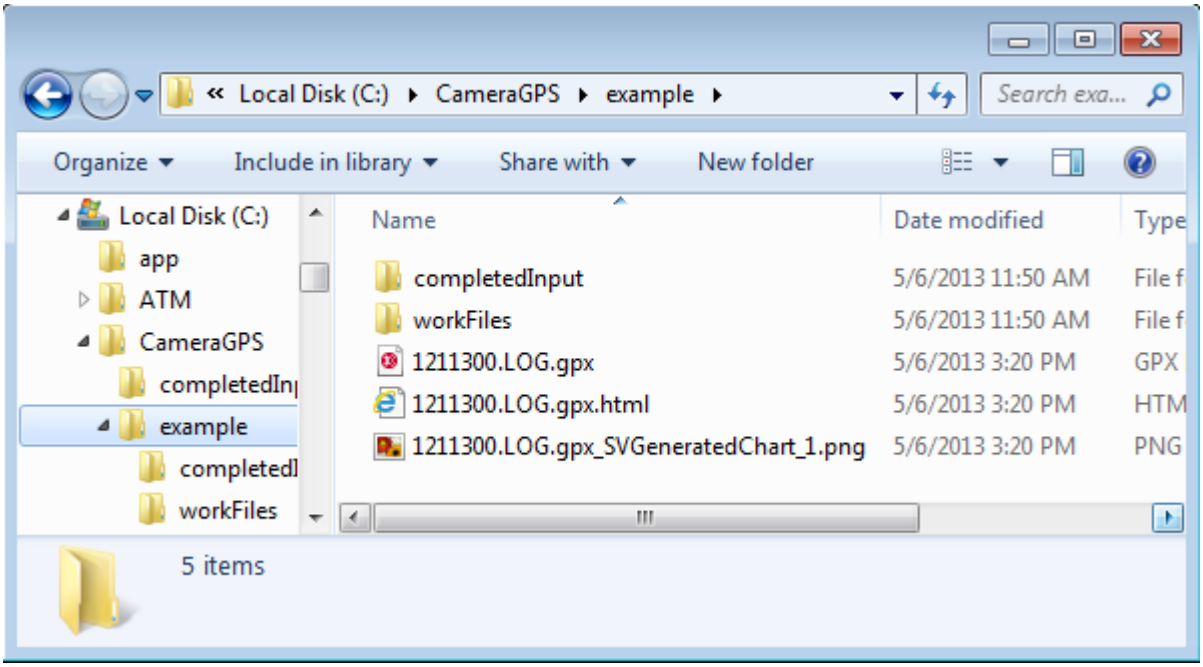
We can drop this file into the C:\CameraGPS\example folder, where it will be processed based on the FlowForce Server job trigger:



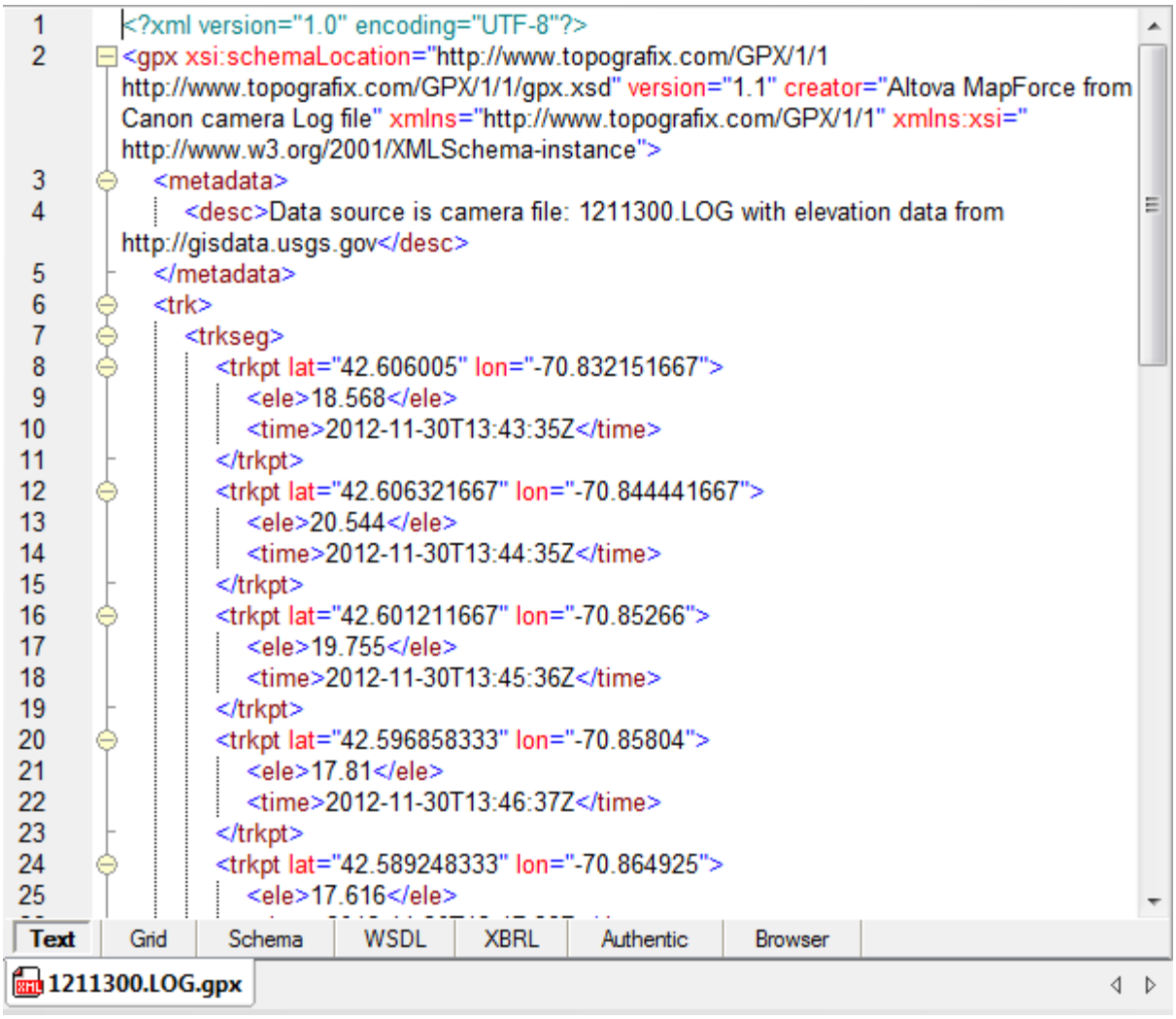
When the timer triggers execution of the FlowForce Server job, the Web interface Job Log page displays this series of messages for the complete job:

Instance	Message
135	Finished job execution: /public/SimpleMapAndTransform
135	Step FlowForce.move completed with status: 0 more
135	Executing FlowForce.move with parameters: {"Source": "C:\\CameraGPS\\example\\1211300.LOG", "Destination": "C:\\CameraGPS\\example\\completedInput", "Working-directory": "", "Overwrite": true}
135	Step StyleVision.TransformationID completed with status: 0 more
	Selected tool: StyleVision 2013r2.
135	Executing StyleVision.TransformationID with parameters: {"OutPdf": "", "InputXml": "C:\\CameraGPS\\example\\1211300.LOG.gpx", "OutDocx": "", "Working-directory": "C:\\CameraGPS\\example\\workFiles", "OutRtf": "", "OutHtml": "C:\\CameraGPS\\example\\1211300.LOG.gpx.html"}
135	Step MapForce.Mapping completed with status: 0 more
	Selected tool: MapForce 2013r2.
135	Executing MapForce.Mapping with parameters: {"Working-directory": "C:\\CameraGPS\\example", "CameraLogFile": "C:\\CameraGPS\\example\\1211300.LOG"}
135	Starting job execution: /public/SimpleMapAndTransform

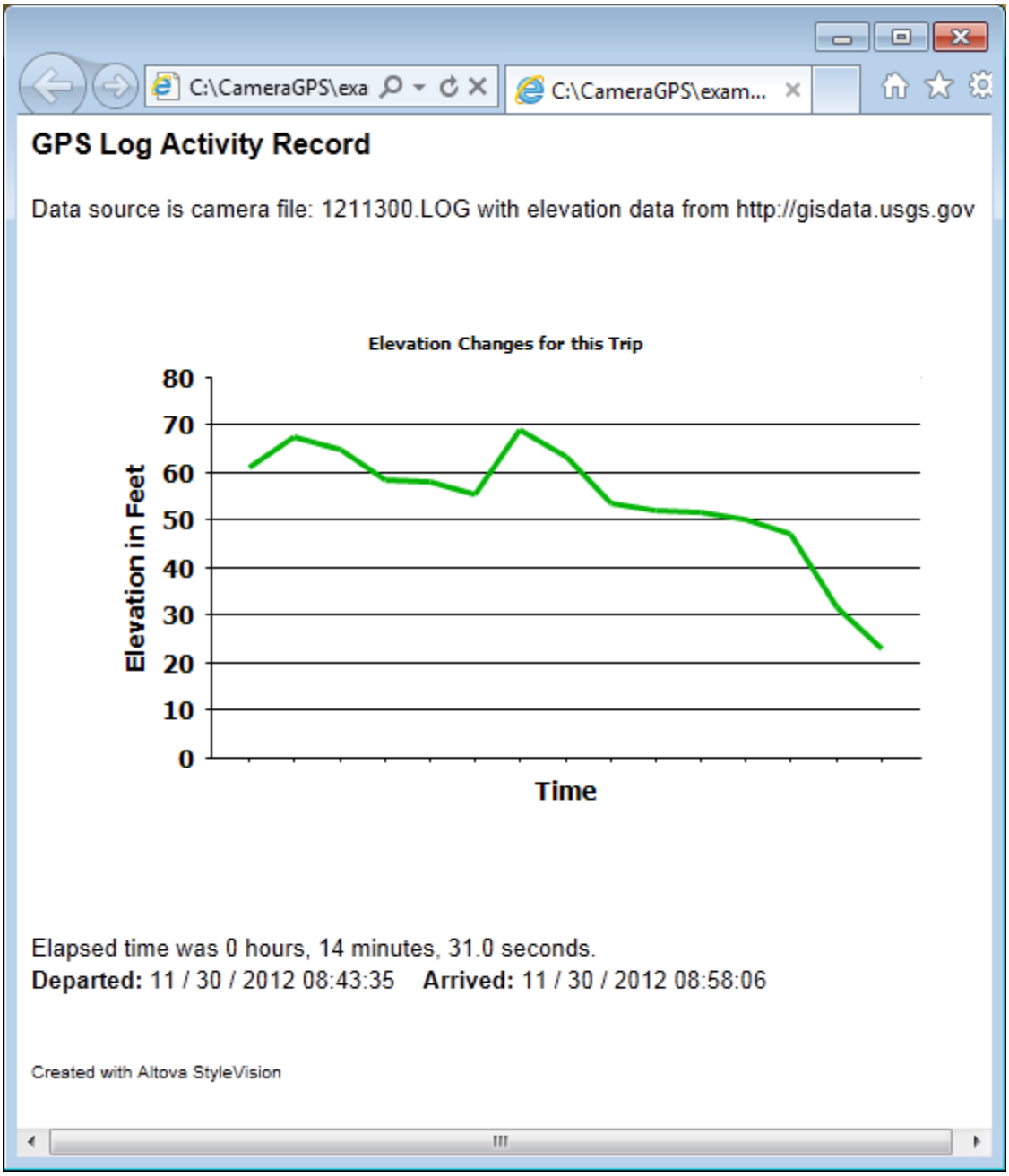
The contents of the C:\CameraGPS\example folder now look like this:



We can examine the 121130.LOG.gpx file in XMLSpy:



And we can open the .html file in any Web browser:



In chapters that follow we will enhance this FlowForce Server example to illustrate jobs with error handling and more complete cleanup of working files.

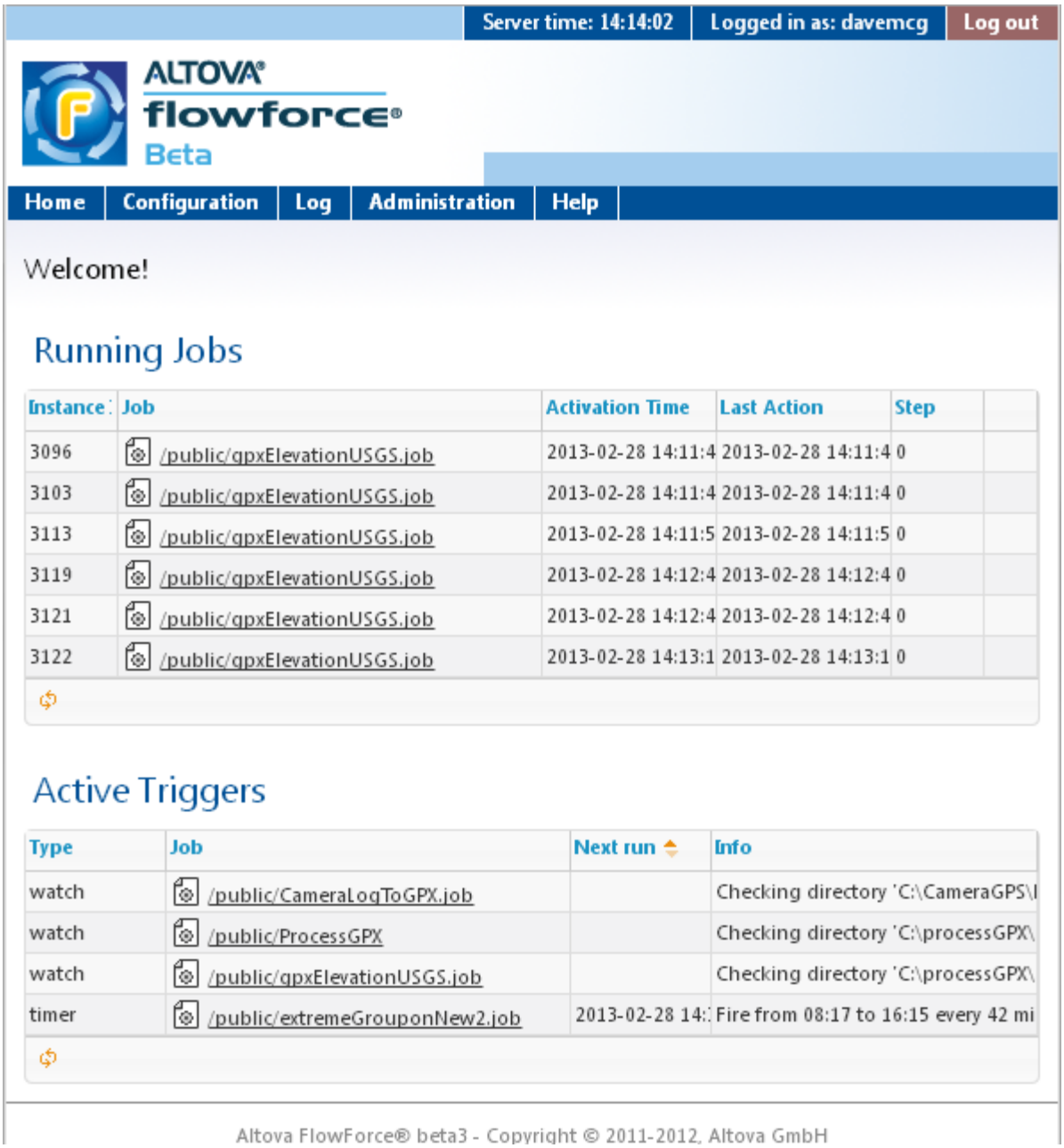
Chapter 5

Web Interface Simplifies Server Management

Multiple users and administrators connect simultaneously via their web browsers

FlowForce Server was designed from the ground up to provide automation of data transformations performed on dedicated high-speed servers. FlowForce Server can start jobs based on a variety of triggers, runs multiple jobs simultaneously, and can even run multiple instances of the same job, depending on workflow.

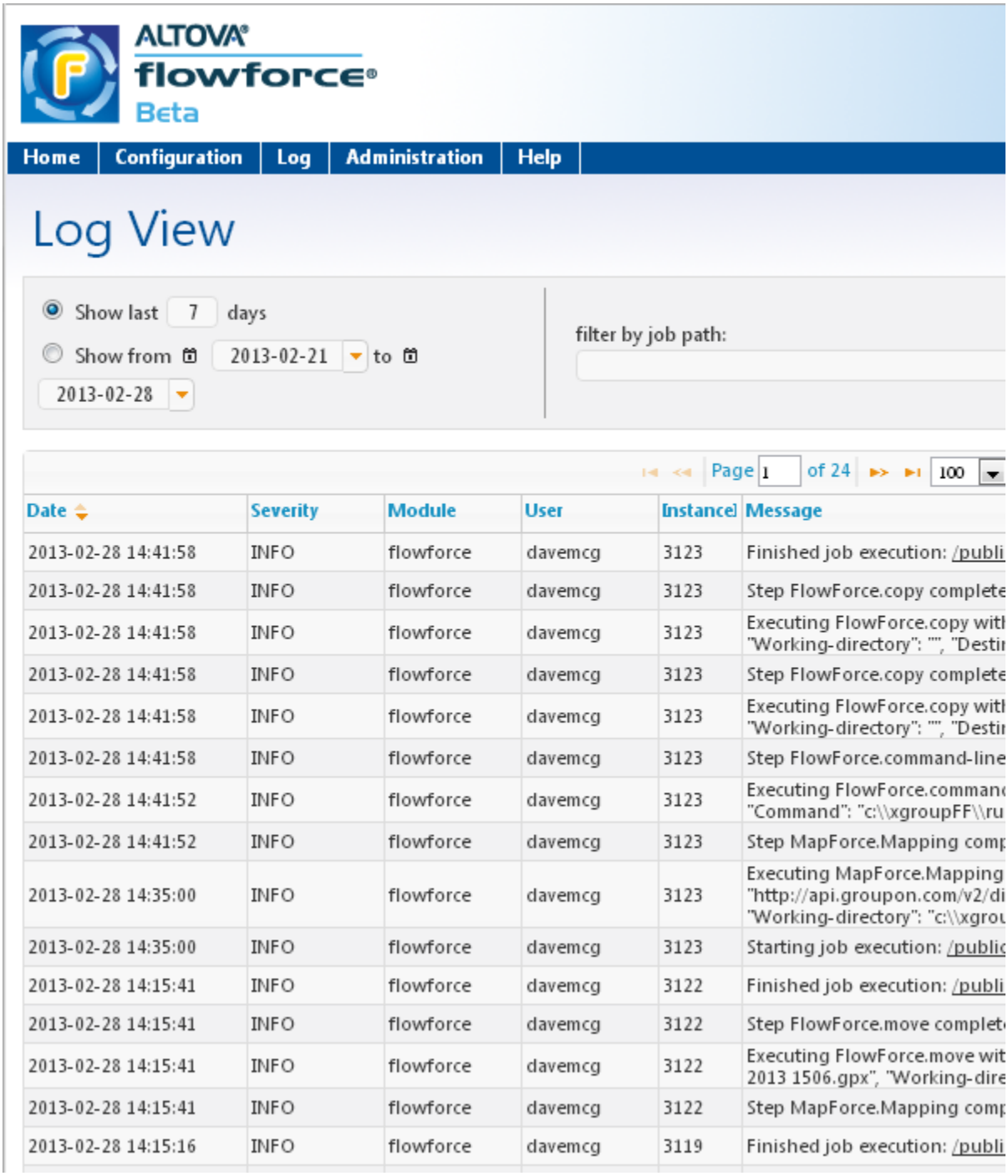
Monitoring all these complex activities is critical to success in a busy production environment. The FlowForce Server Web interface includes customizable views into operations, simplifying management from anywhere on the network.



The screen shot above shows the Home page of the FlowForce Server Web browser interface, displaying all currently running jobs and active triggers. Six instances of the gpxElevationUSGS job are running, each identified by a unique job ID. Four job triggers are also active, three watching hot folders, and one based on a timer. Each FlowForce Server job automates a MapForce data transformation.

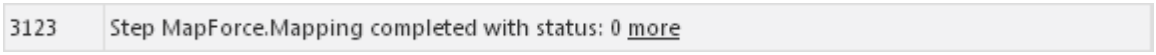
The orange arrows below each grid are clickable update buttons, and the job names link to the definition pages for each job.

The blue headings at the top are also clickable buttons. The Log link displays the Log View, a detailed history of all system activity, shown in a truncated version below.



Date	Severity	Module	User	Instance	Message
2013-02-28 14:41:58	INFO	flowforce	davemcg	3123	Finished job execution: /publi
2013-02-28 14:41:58	INFO	flowforce	davemcg	3123	Step FlowForce.copy complete
2013-02-28 14:41:58	INFO	flowforce	davemcg	3123	Executing FlowForce.copy with "Working-directory": "", "Destin
2013-02-28 14:41:58	INFO	flowforce	davemcg	3123	Step FlowForce.copy complete
2013-02-28 14:41:58	INFO	flowforce	davemcg	3123	Executing FlowForce.copy with "Working-directory": "", "Destin
2013-02-28 14:41:58	INFO	flowforce	davemcg	3123	Step FlowForce.command-line
2013-02-28 14:41:52	INFO	flowforce	davemcg	3123	Executing FlowForce.comman "Command": "c:\\xgroupFF\\ru
2013-02-28 14:41:52	INFO	flowforce	davemcg	3123	Step MapForce.Mapping comp
2013-02-28 14:35:00	INFO	flowforce	davemcg	3123	Executing MapForce.Mapping "http://api.groupon.com/v2/di "Working-directory": "c:\\xgrou
2013-02-28 14:35:00	INFO	flowforce	davemcg	3123	Starting job execution: /public
2013-02-28 14:15:41	INFO	flowforce	davemcg	3122	Finished job execution: /publi
2013-02-28 14:15:41	INFO	flowforce	davemcg	3122	Step FlowForce.move complete
2013-02-28 14:15:41	INFO	flowforce	davemcg	3122	Executing FlowForce.move wit 2013 1506.gpx", "Working-dire
2013-02-28 14:15:41	INFO	flowforce	davemcg	3122	Step MapForce.Mapping comp
2013-02-28 14:15:16	INFO	flowforce	davemcg	3119	Finished job execution: /publi

Each message line describes one step in one instance of a FlowForce Server job, and may have links to more information, as we described the previous charpter. If the status is anything other than 0, the more link opens a detailed error message.



In a busy environment, all this can be just too much information. In that case, the Log View offers several alternatives to help you find critical information quickly. The options bar above the list lets you filter items displayed by a particular job or Message severity.

If you only want to know if anything unexpected occurred, select Warning as shown above to hide all normal Info messages.

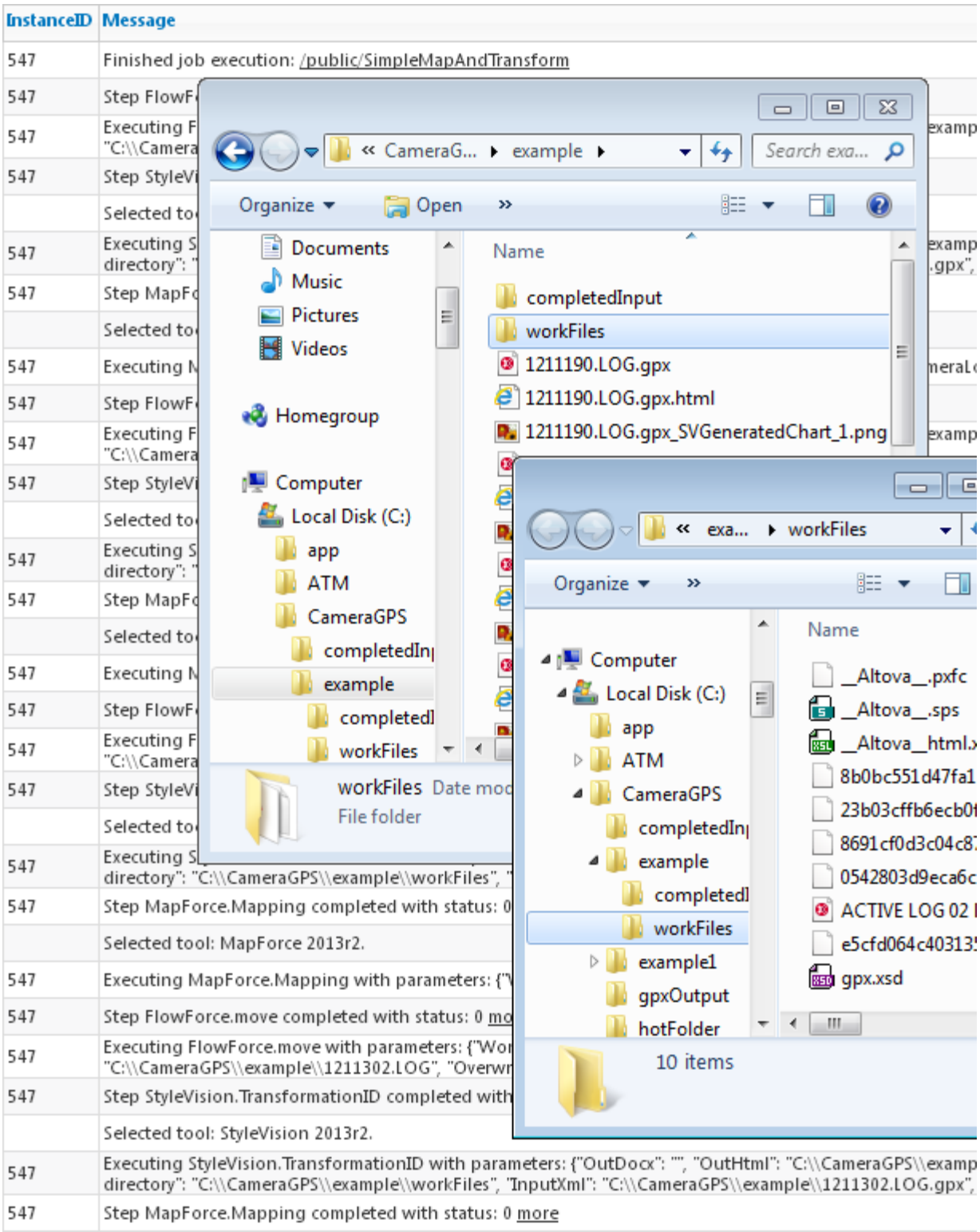
You can also sort the list based on the column headings Date, Severity, Module, User, or Instance ID in ascending or descending order. When the FlowForce Server is running multiple simultaneous jobs and instances, it's very likely individual steps from different jobs will be shuffled. Sorting by Instance ID can let you more easily review the job history.

Chapter 6

Customizing a FlowForce Server Job

Automate file management and other housekeeping tasks on a busy server

In our earlier chapter titled Automate Data Transformation with FlowForce Server, we created a job called SimpleMapAndTransform to automate data mapping with MapForce Server and creation of html reports by StyleVision Server. After the FlowForce Server job ran several times, we have accumulated many output files in the same folder we use to process input files, as well as temporary intermediate files in the workFiles folder, as seen in the image below.

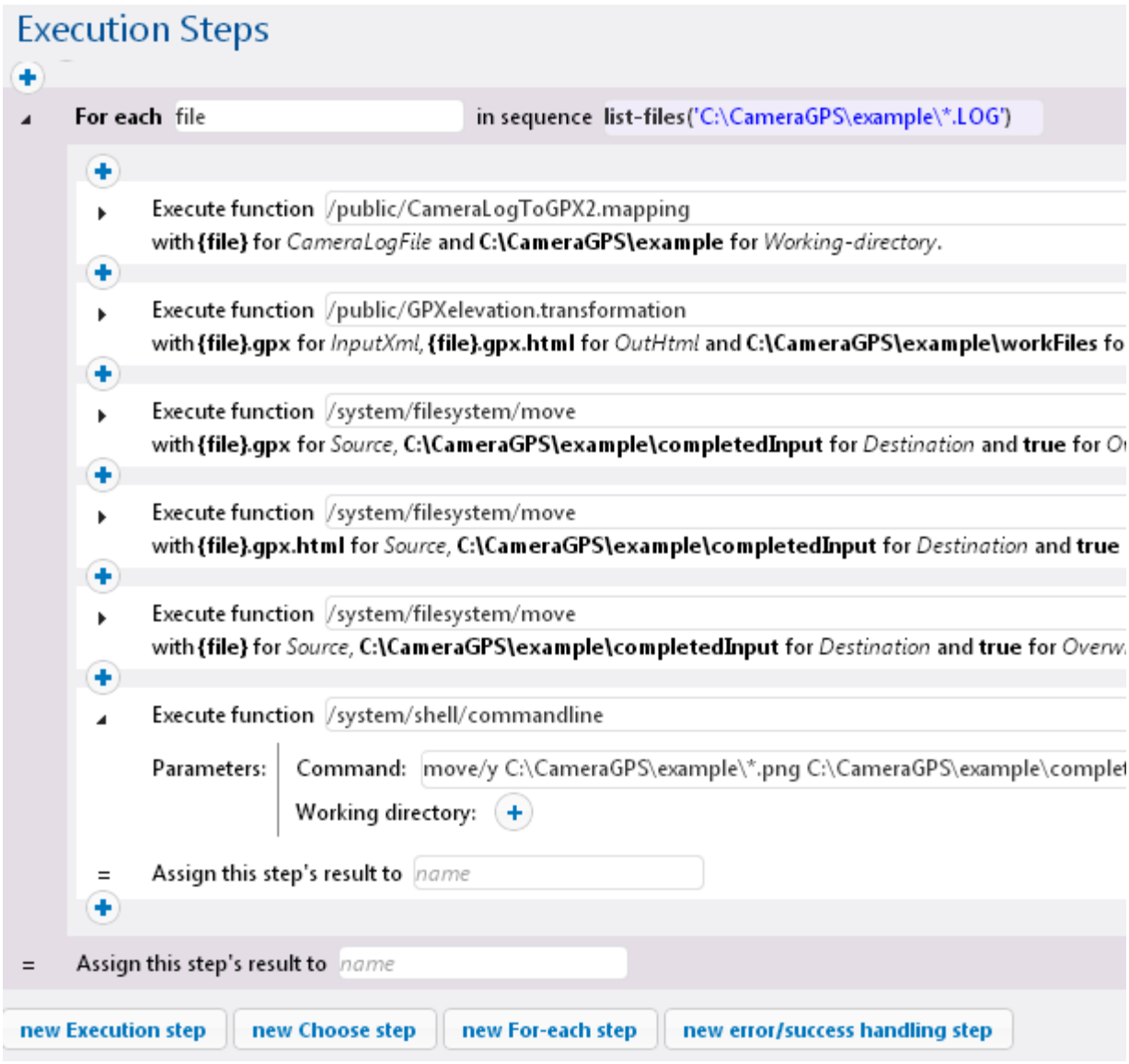


In this chapter we will enhance the job to create more orderly results and remove unneeded temporary files.

Reorganizing Output

First, we can add additional job steps inside the **For each** loop that processes input files to move the new .gpx and .html files created by the data mapping and transformation to the completedInput folder. These new steps are third and fourth in the series below.

The Working-directory option in each step can also redirect output. However, the {file} variable we used to select each input file for processing contains the full path and the file name, so it's simpler to just move all the output files when processing is complete.



Our StyleVision design for the report includes a chart that is saved as a .png file separately from the main .html document. The last step in the job uses a system

commandline function to move all .png files from the input folder to the completed work folder.

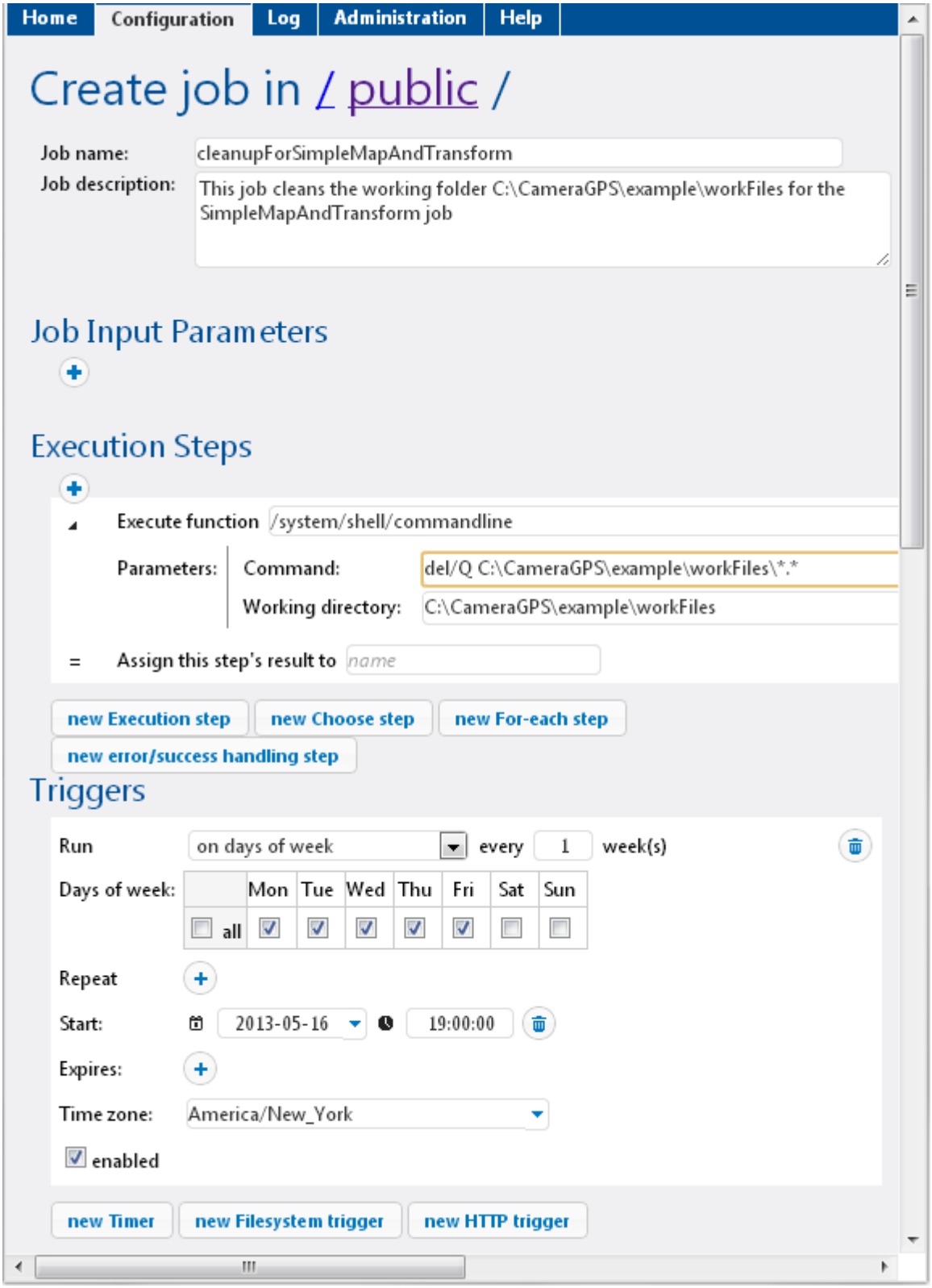
The commandline function lets you define any valid operating system command as a FlowForce Server job step. In this job we are still inside the For-each loop, so the .png files will move along with the associated .html document. We used a wildcard for the .png filename because the .png files do not precisely follow the naming pattern of the input documents. The wildcard pattern also works for StyleVision designs that create multiple charts for each report.

Housekeeping Job Steps

It’s good practice to clear the contents of working directories periodically. Our job uses the workFiles folder to expand the contents of the .pxf file for report rendering. The screenshot of this directory in the image at the top of this post shows the XML Schema, working files for the design, .XSLT files to render various report formats, and temporary working files.

We could add a housekeeping step after the for-Each loop in our job to clean up immediately after processing. However, FlowForce Server lets you run multiple simultaneous instances of the same job. It would be unfortunate for one instance to clear the workFiles folder while another instance is still using it!

A better solution is to define a separate housekeeping job and schedule it to run at a time when the main job is idle. Here is a shot of a very simple cleanup job for the workingFiles folder, applying the commandline function again:

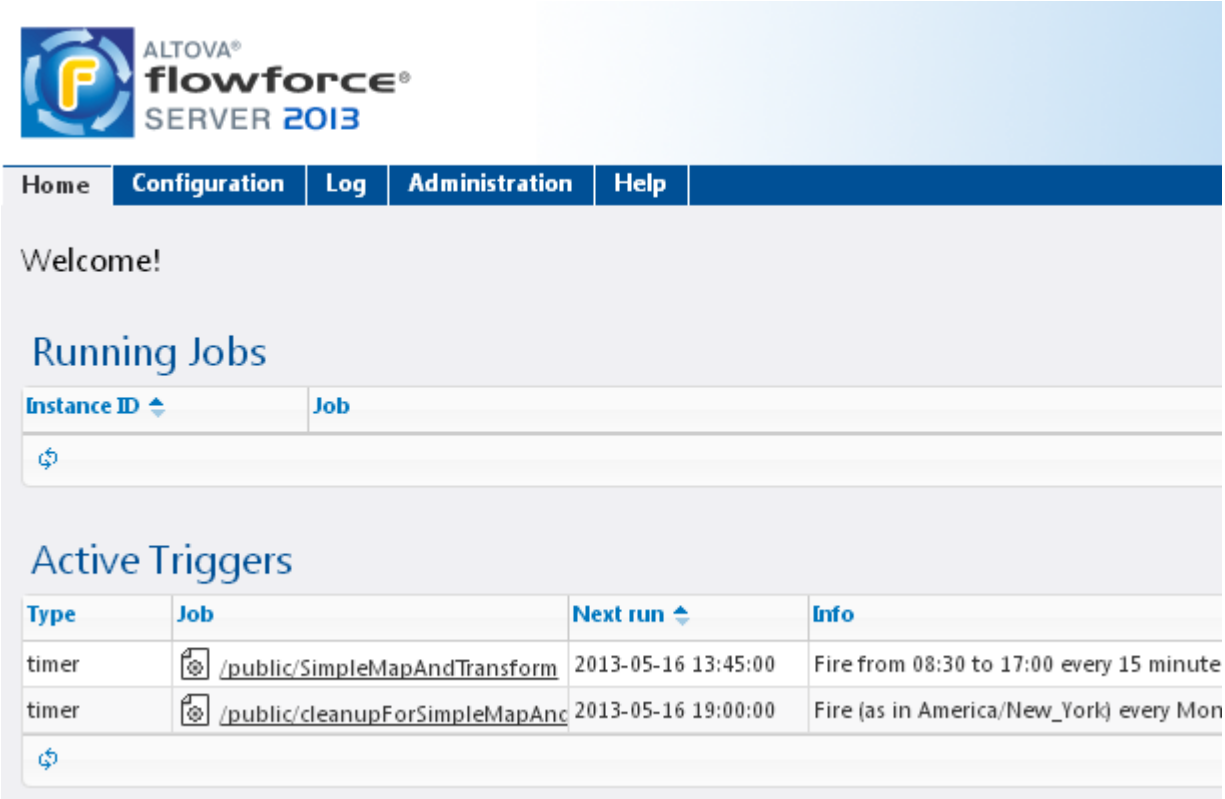


We used a wildcard to delete everything in the workFiles directory, but that option may not be appropriate in all cases. If the .pxf file contains .css stylesheets or image files

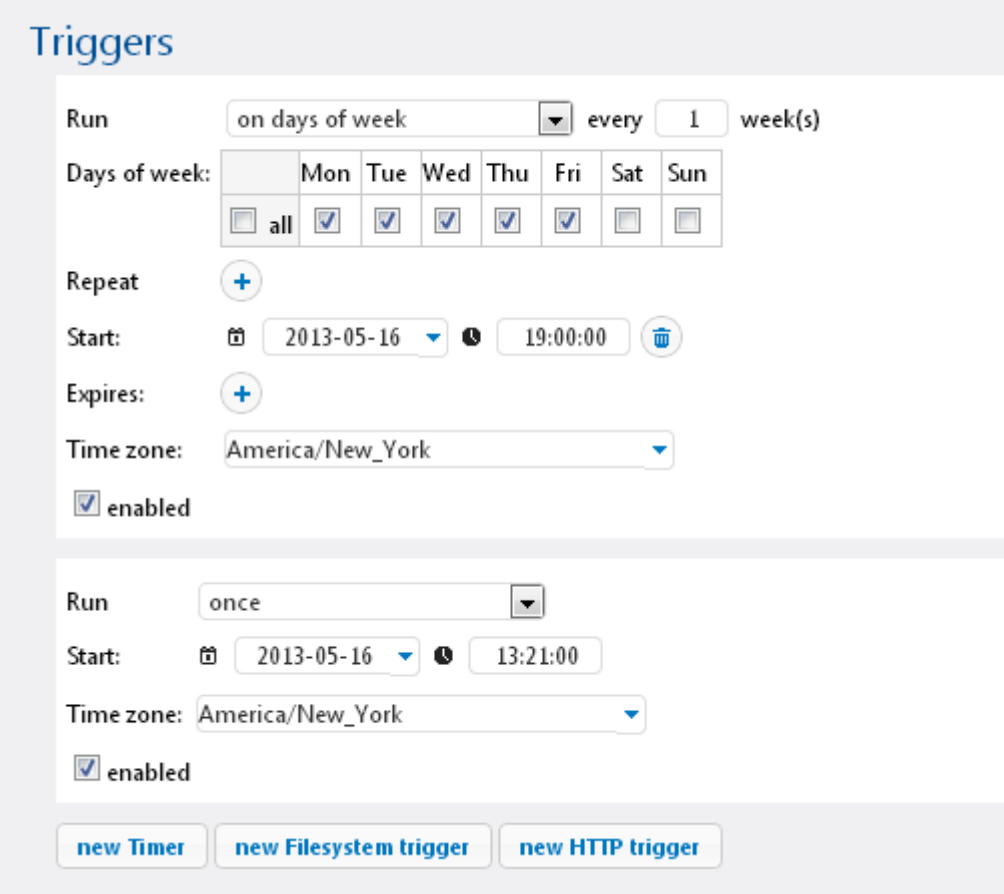
that are needed by the .html document, more customized housekeeping job steps will be needed.

Job Triggers

The original SimpleMapAndTransform job is scheduled to run every 15 minutes from 8:30 to 5:00, Monday through Friday. As shown above, the cleanup job is scheduled to run once per day at 19:00, or 7:00 PM. The FlowForce Server Web interface home page shows the job schedule:



Here's a quick trick you can use to test a FlowForce Server job as you define it. Every job can have multiple calendar triggers. We can define a run-once trigger to immediately test a new job and schedule it for just a few seconds in the future:



We can get immediate feedback by reading the FlowForce Server Log for the details of each job step.

Chapter 7

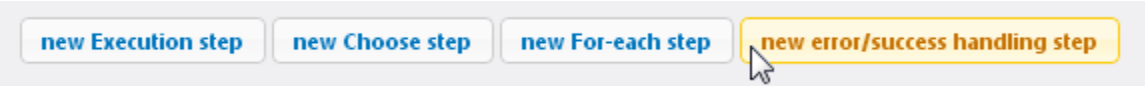
Taming Bad Input Data with FlowForce Server

Recover from errors in input files and keep production flowing

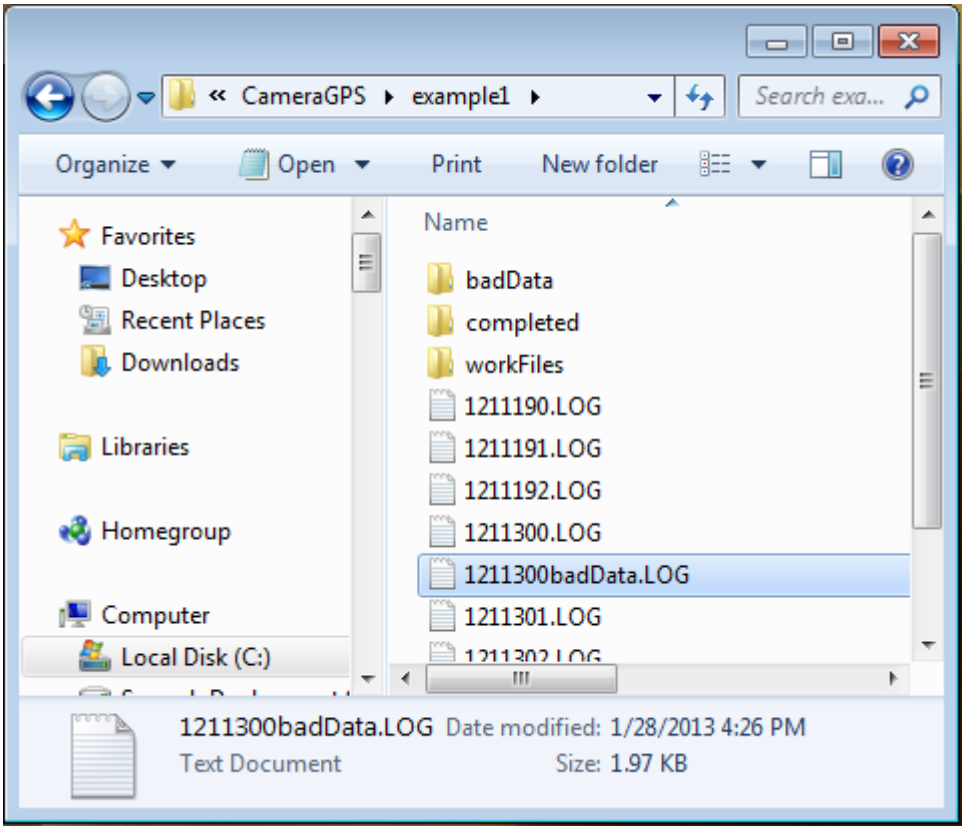
Whenever you accept data from an outside source you risk encountering errors. We wrote in the Altova Blog about this phenomenon in the past in Expect the Unexpected – Altova MissionKit Solves a Number Format Mystery and in the series of posts on Processing the Groupon API.

Bad data in an input file can cause the data transformation step of a FlowForce Server job to fail. When a FlowForce Server Job fails, further execution steps will not be performed. FlowForce Server is designed this way to prevent an error in one job step from cascading into a series of additional invalid results. Happily, FlowForce Server also includes features to help you recover from errors and keep production flowing.

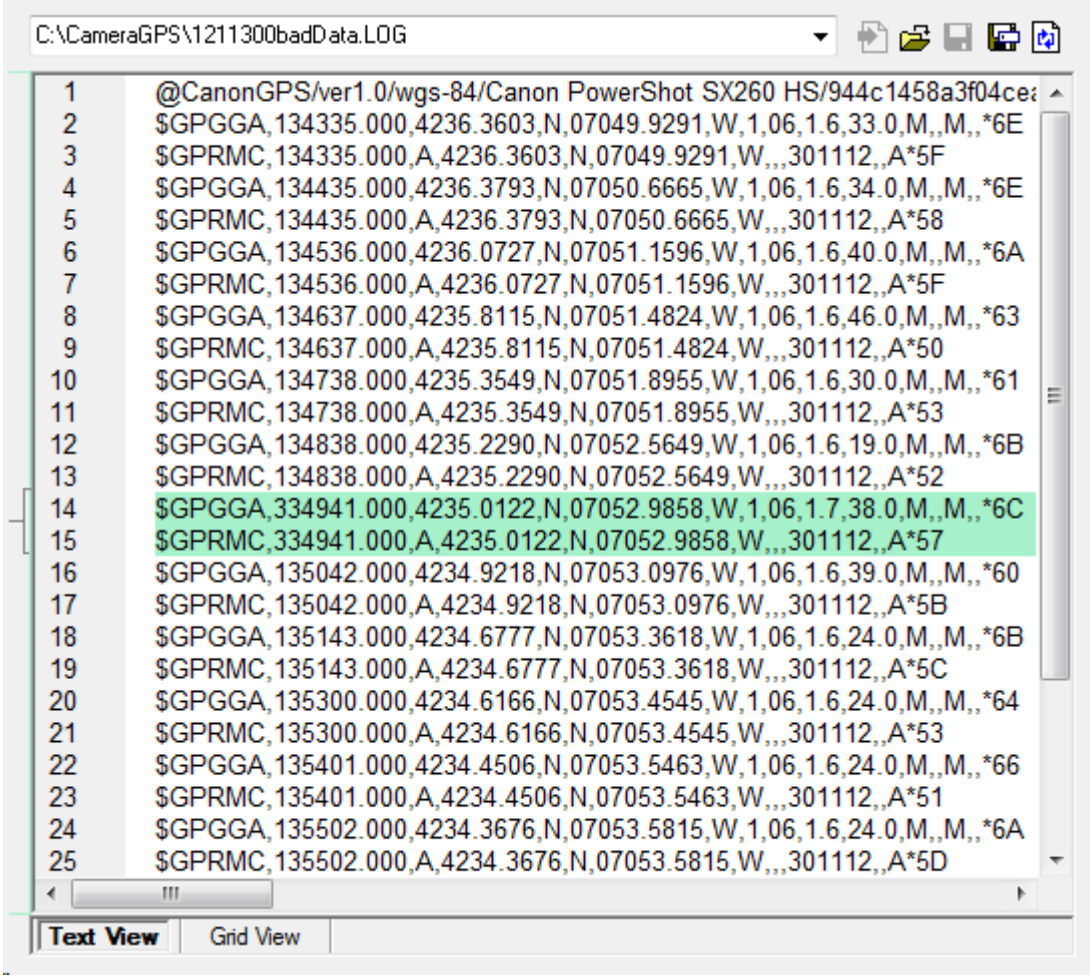
In this chapter we will further extend the data mapping and report rendering job described in Customizing a FlowForce Server Job to gracefully handle bad data in an input file.



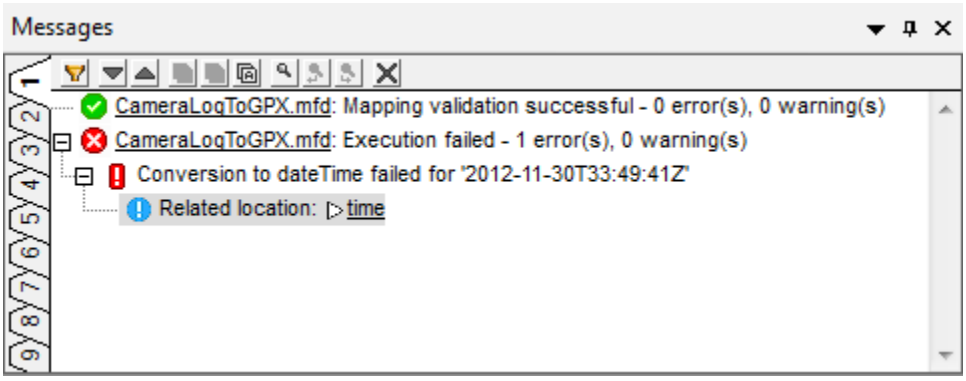
We started by creating a variation of one of the files that was edited to include invalid data, and we added a folder to the workflow to be the destination for bad input files.



The first numeric column in the input .csv file is a time stamp for hours, minutes, seconds, and thousandths. We opened the file in DiffDog and simply edited the values on lines 14 and 15 to be outside the 24-hour maximum.



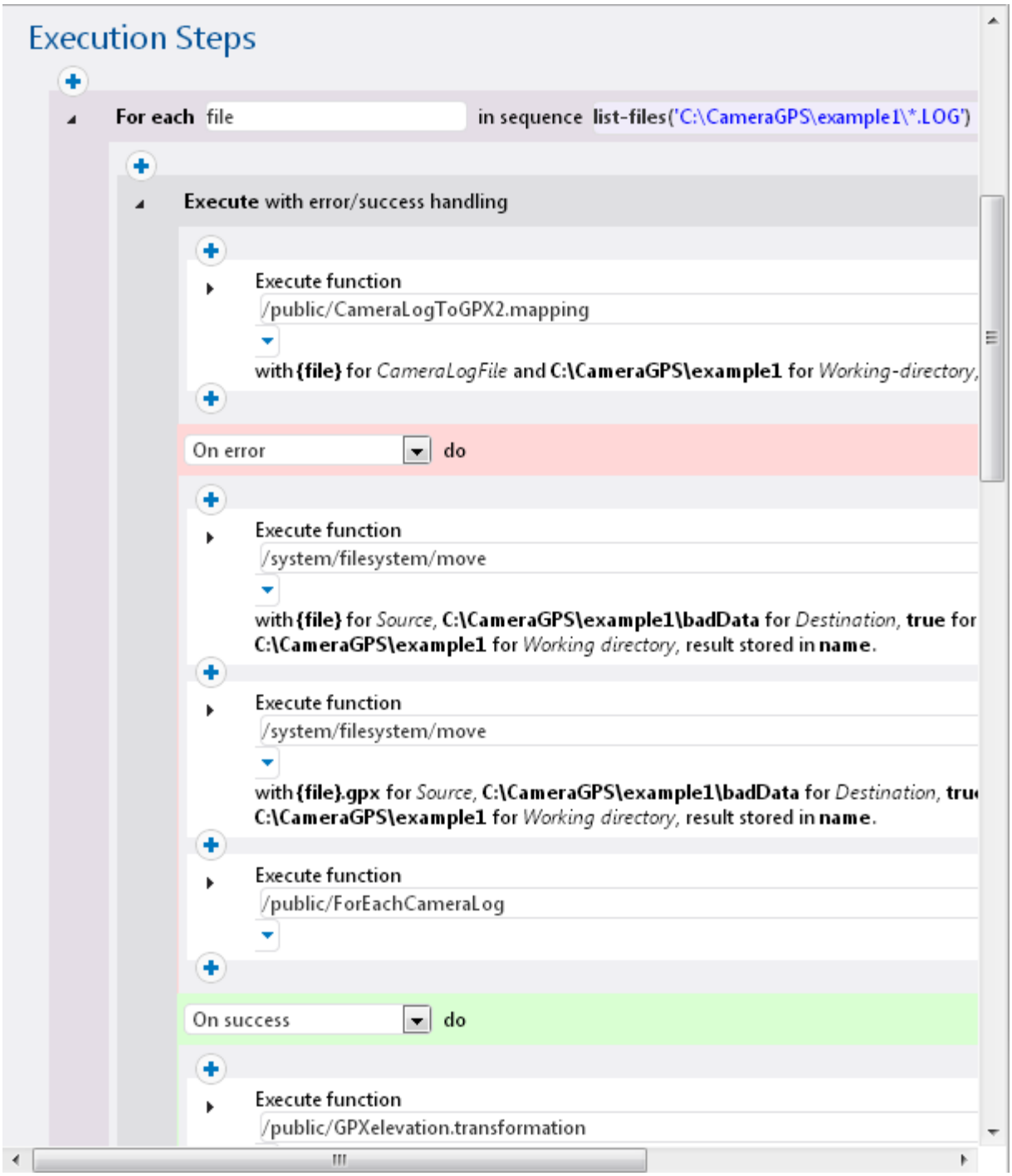
As an initial test of the bad data file, we launched MapForce and assigned it as the input for the CameraLogToGPX mapping. When we clicked the Output button to execute the mapping, the following error occurred:



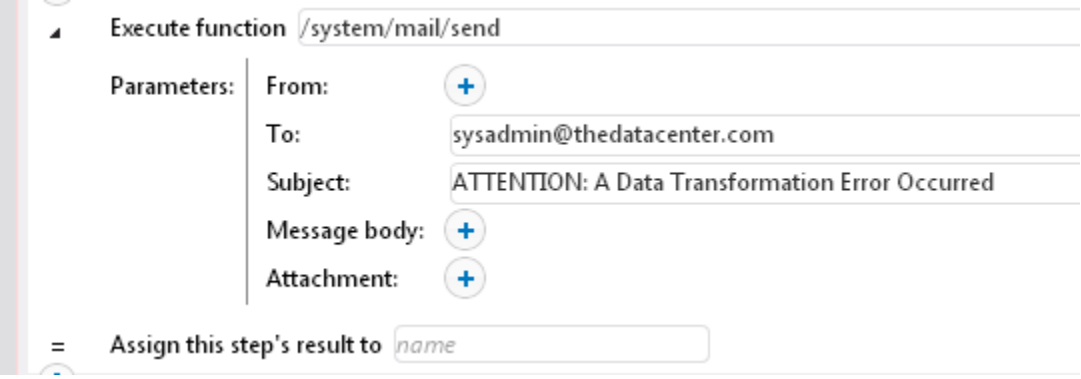
FlowForce Server Job Steps with Error Handling

Next, we defined a new version of the FlowForce Server job to process input data files inside an **Error/Success Handling** step. If the data mapping fails, we will move the bad input file and any partially written output .gpx file to the badData folder. If the data mapping is successful, we continue on to do the transformation to generate the .html report, then move the input and .gpx files to the completed work folder.

Note the last step of the **On-error** section. A data mapping error will halt execution of the For-each file loop, so we recursively call the whole job again to finish any unprocessed input files.



If a data mapping error is critical to the enterprise and demands immediate action, we could even add a job step inside the On-error section to send an email message:



Of course the addressee, Subject, Massage Body, and Attachment fields are fully configurable.

If the data mapping step succeeds, FlowForce Server executes the **On-success** section and performs the .html rendering job step.

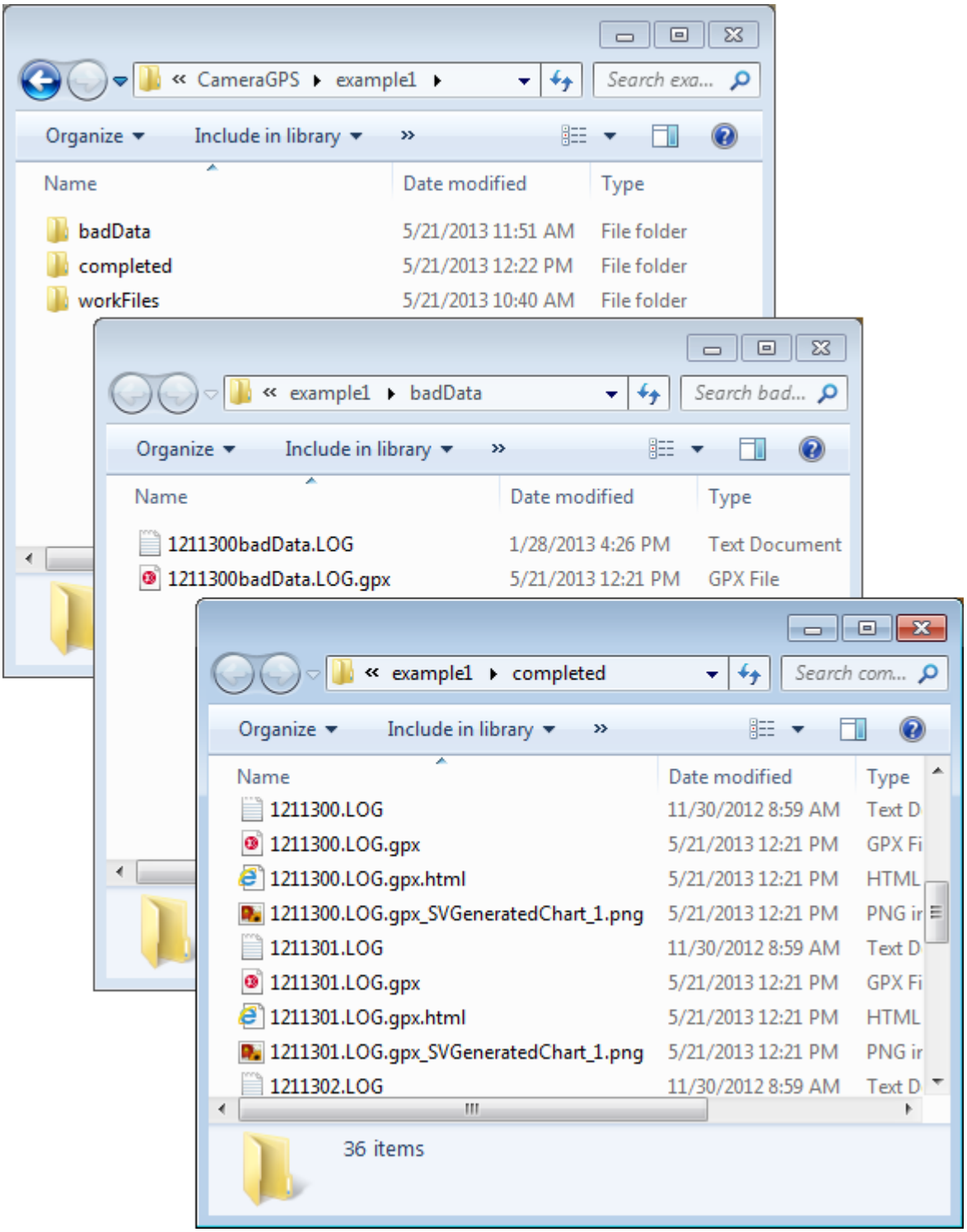
Running the Job

The job fires based on a time trigger, and the FlowForce Server Log records each execution step. In the portion of the log below we can see how the bad data input file is handled. (The Log is sorted with the oldest entry first.) The third entry in the sequence indicates the error. Next, the input file and partially-generated .gpx file are sent to the badData folder.

2013-05-21 12:21:36	INFO	715	Executing MapForce.Mapping with parameters: {"Working-directory": "C:\\CameraGPS\\example1", "CameraLogFile": "C:\\CameraGPS\\example1\\1211300badData.LOG"}
2013-05-21 12:21:36	INFO		Selected tool: MapForce 2013r2.
2013-05-21 12:21:39	ERROR	715	Step MapForce.Mapping completed with status: 1 more
2013-05-21 12:21:39	INFO	715	Executing FlowForce.move with parameters: {"Overwrite": true, "Destination": "C:\\CameraGPS\\example1\\badData", "Source": "C:\\CameraGPS\\example1\\1211300badData.LOG", "Working-directory": "C:\\CameraGPS\\example1"}
2013-05-21 12:21:40	INFO	715	Step FlowForce.move completed with status: 0 more
2013-05-21 12:21:40	INFO	715	Executing FlowForce.move with parameters: {"Overwrite": true, "Destination": "C:\\CameraGPS\\example1\\badData", "Source": "C:\\CameraGPS\\example1\\1211300badData.LOG.gpx", "Working-directory": "C:\\CameraGPS\\example1"}
2013-05-21 12:21:40	INFO	715	Step FlowForce.move completed with status: 0 more
2013-05-21 12:21:41	INFO	715	Executing MapForce.Mapping with parameters: {"Working-directory": "C:\\CameraGPS\\example1", "CameraLogFile": "C:\\CameraGPS\\example1\\1211301.LOG"}

The last entry above shows the server starting work on the next input file in the folder.

When the job completes processing, we see the expected results in the contents of the working folders:



Chapter 8

FlowForce Server Supports RaptorXML

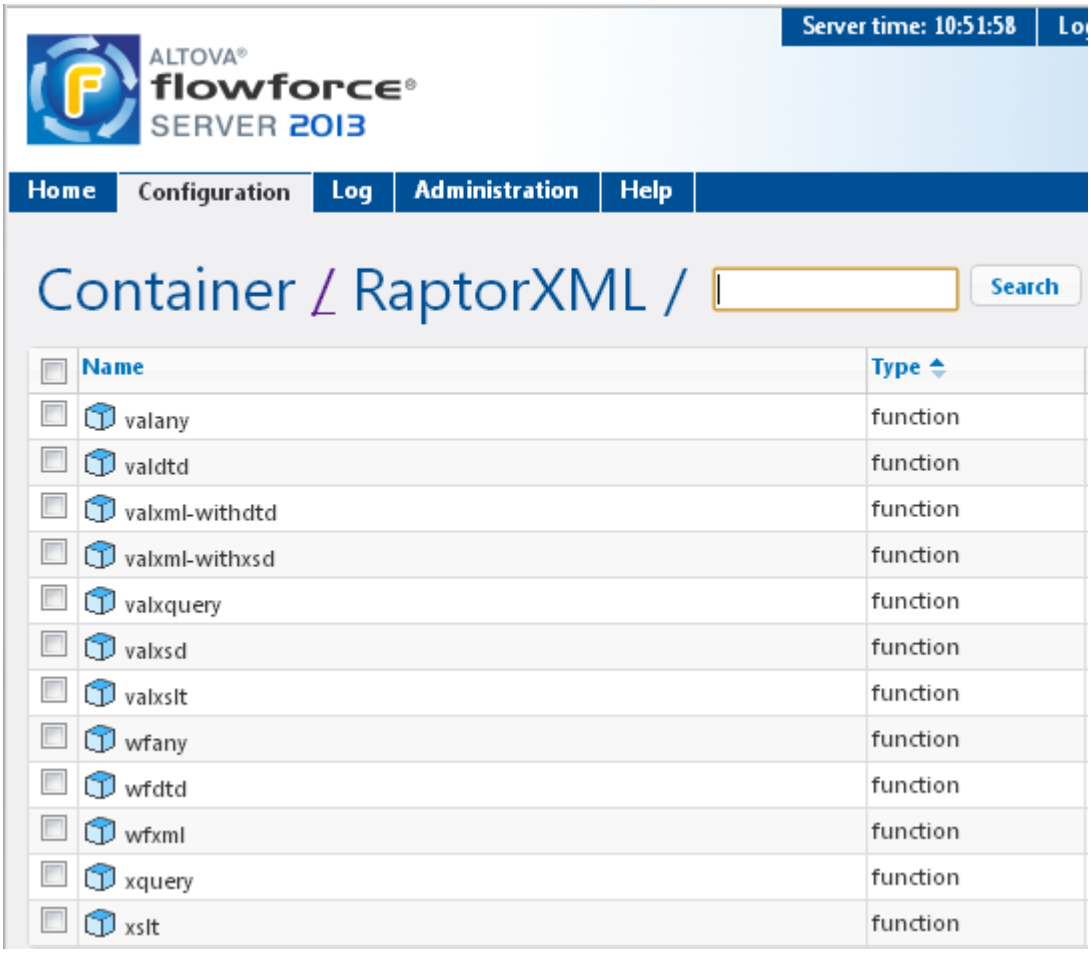
Validate XML and XBRL, perform XSLT transformations, execute XQuery, and more

Altova FlowForce Server supports RaptorXML Server and RaptorXML+XBRL Server.

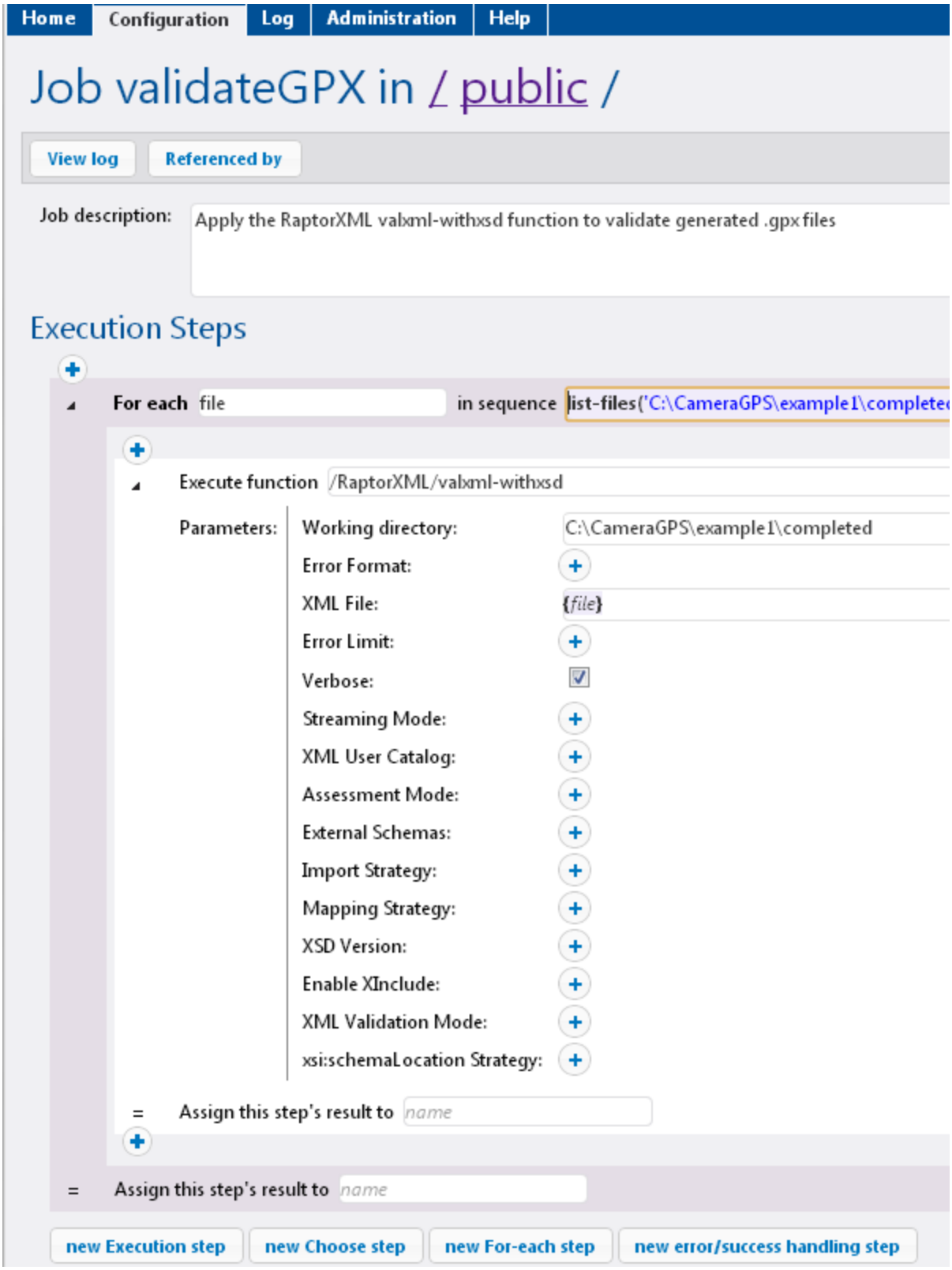
Altova RaptorXML is the third-generation, hyper-fast XML and XBRL processor from the makers of XMLSpy. RaptorXML is built from the ground up to be optimized for the latest standards and parallel computing environments. Now FlowForce Server jobs can include steps to validate XML, transform XML with XSLT, execute XPath and XQuery, and even perform complex XBRL operations for financial reporting including validating XBRL taxonomies, and validating XBRL instances against XBRL taxonomies with support for XBRL Dimensions, XBRL Formula, and XBRL Table Linkbase, which define new, compatible functionality to extend XBRL 2.1.

For example, we can return to the FlowForce Server job described in Taming Bad Input Data with FlowForce Server and add a RaptorXML operation to validate the XML files created from raw GPS data captured by a digital camera.

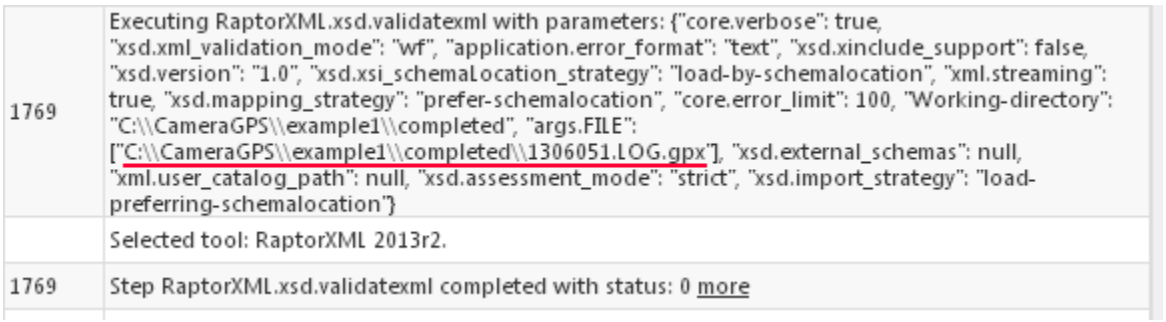
RaptorXML Server provides specialized functions to validate XML, check well-formedness, and perform XQuery and XSLT operations.



We can create a new job to validate the .gpx files generated by our earlier FlowForce Server job in the completed work folder. We can run the validation job in standalone mode for testing, and when we are satisfied with the results, add it as a new step to the original job.

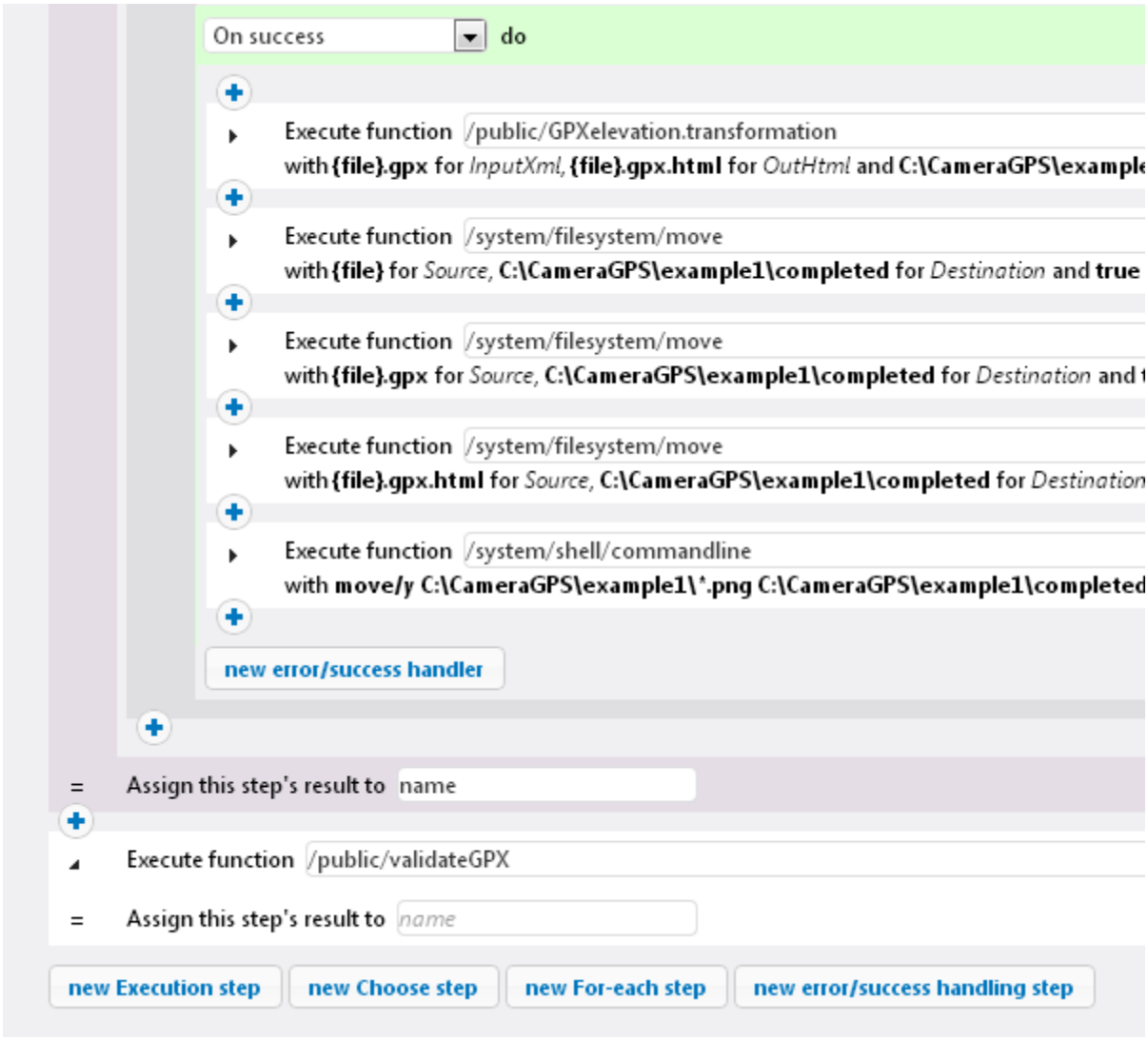

Home
Configuration
Log
Administration
Help
Job validateGPX in / public /
View log
Referenced by
Job description:
Apply the RaptorXML valxml-withxsd function to validate generated .gpx files
Execution Steps
For each
file
in sequence
list-files('C:\CameraGPS\example1\completed
Execute function
/RaptorXML/valxml-withxsd
Parameters:
Working directory:
C:\CameraGPS\example1\completed
Error Format:
XML File:
{file}
Error Limit:
Verbose:
Streaming Mode:
XML User Catalog:
Assessment Mode:
External Schemas:
Import Strategy:
Mapping Strategy:
XSD Version:
Enable XInclude:
XML Validation Mode:
xsi:schemaLocation Strategy:
=
Assign this step's result to
name
=
Assign this step's result to
name
new Execution step
new Choose step
new For-each step
new error/success handling step

When the job runs, each .gpx file in the completed work folder is validated, as shown in the portion of the FlowForce Server Log shown below. We added a red underline to the illustration to highlight the file name for this instance.



1769	Executing RaptorXML.xsd.validatexml with parameters: {"core.verbose": true, "xsd.xml_validation_mode": "wf", "application.error_format": "text", "xsd.xinclude_support": false, "xsd.version": "1.0", "xsd.xsi_schemaLocation_strategy": "load-by-schemalocation", "xml.streaming": true, "xsd.mapping_strategy": "prefer-schemalocation", "core.error_limit": 100, "Working-directory": "C:\\CameraGPS\\example1\\completed", "args.FILE": ["C:\\CameraGPS\\example1\\completed\\1306051.LOG.gpx"], "xsd.external_schemas": null, "xml.user_catalog_path": null, "xsd.assessment_mode": "strict", "xsd.import_strategy": "load-preferring-schemalocation"}
	Selected tool: RaptorXML 2013r2.
1769	Step RaptorXML.xsd.validatexml completed with status: 0 more

Now we can easily include the validation job as a new step in the original job, placing it at the end to check all completed work:


On success do
Execute function /public/GPXelevation.transformation
with {file}.gpx for InputXml, {file}.gpx.html for OutHtml and C:\CameraGPS\example
Execute function /system/filesystem/move
with {file} for Source, C:\CameraGPS\example1\completed for Destination and true
Execute function /system/filesystem/move
with {file}.gpx for Source, C:\CameraGPS\example1\completed for Destination and
Execute function /system/filesystem/move
with {file}.gpx.html for Source, C:\CameraGPS\example1\completed for Destination
Execute function /system/shell/commandline
with move/y C:\CameraGPS\example1*.png C:\CameraGPS\example1\completed
new error/success handler
= Assign this step's result to name
Execute function /public/validateGPX
= Assign this step's result to name
new Execution step
new Choose step
new For-each step
new error/success handling step

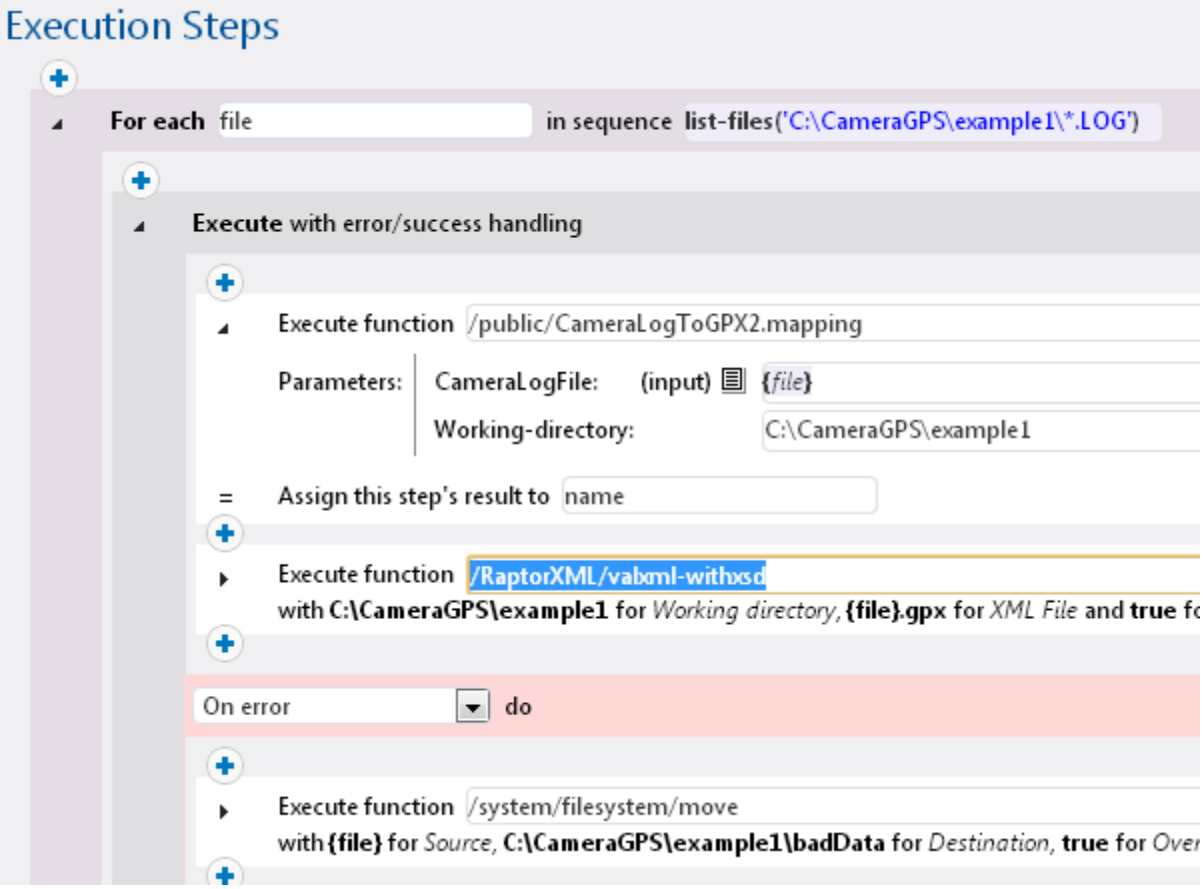
Chapter 9

Constant Quest for Efficiency

Get the job done in the fewest steps

In the previous chapter, we created a FlowForce Server job that defined a RaptorXML execution step to validate XML files, and we called that job as a step at the end of our camera GPS job, as a final check on the output. That was a quick way to demonstrate integration of FlowForce Server and RaptorXML Server, but for real-world production we would want to perform the same task more efficiently.

If we insert the RaptorXML validation function at the top of the job, right in front of the ***On error*** definition as shown below, we can apply the same error handling steps for failures of either the data mapping or the validation step. In other words, an error in any one of a series of steps before **On error** forces the job to take the error path.

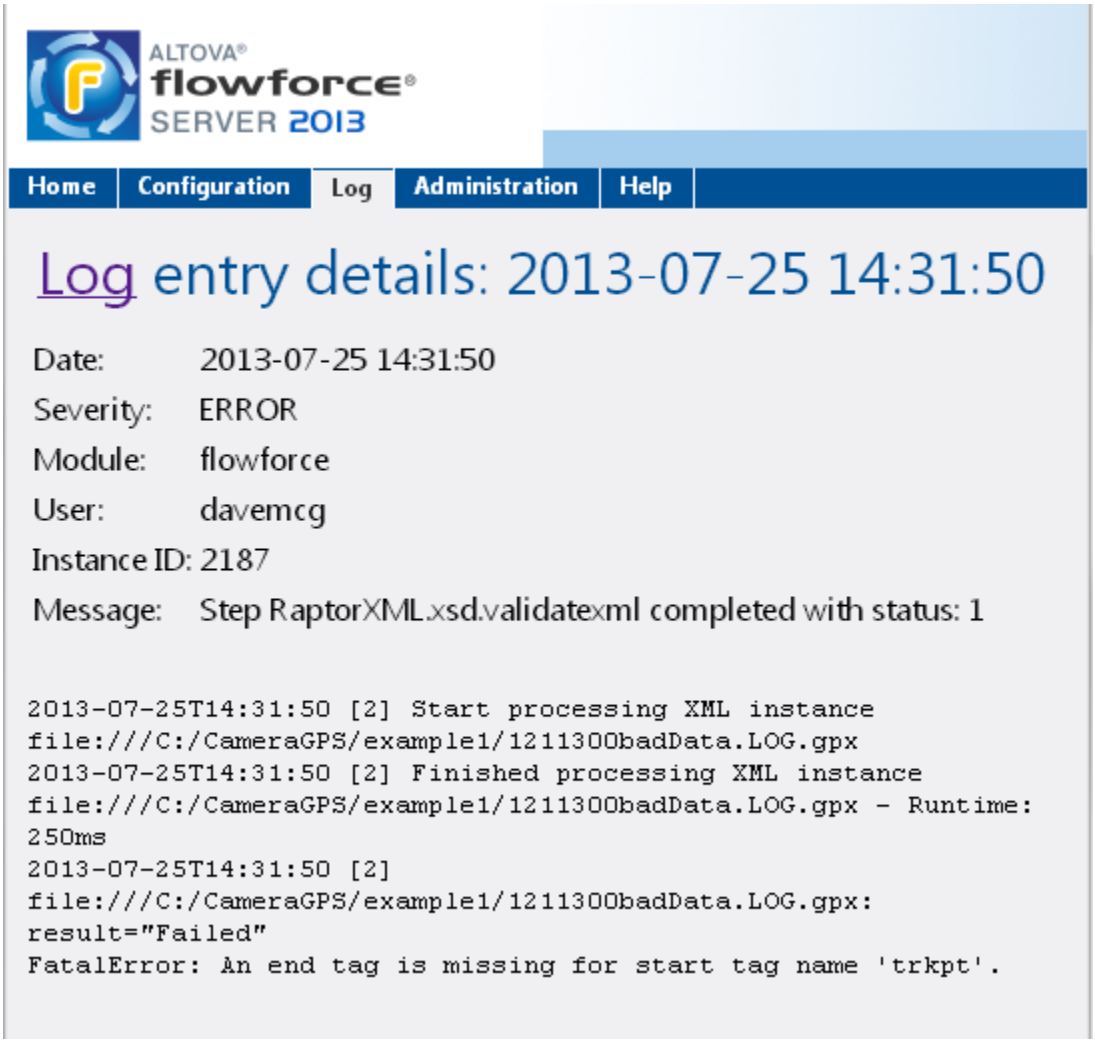


If by some chance the mapping was successful but the .gpx output file was not valid, catching it here instead of at the very end of our original job avoids sending bad data on to StyleVision Server for report generation.

When we run the revised job and encounter an input file with bad data, the mapping function stops immediately and the validation step fails as well, indicated on line 3 and line 6 of the partial FlowForce Server Log view shown below where job steps “completed with status: 1.”

2187	Step FlowForce.command-line completed with status: 0 more
2187	Executing MapForce.Mapping with parameters: {"Working-directory": "C:\\CameraGPS\\example1", "CameraLogFile": "C:\\CameraGPS\\example1\\1211300badData.LOG"}
	Selected tool: MapForce 2013r2.
2187	Step MapForce.Mapping completed with status: 1 more
2187	Executing RaptorXML.xsd.validatexml with parameters: {"core.verbose": true, "xsd.xml_validation_mode": "wf", "application.error_format": "text", "xsd.xinclude_support": false, "xsd.version": "1.0", "xsd.xsi_schemaLocation_strategy": "load-by-schemalocation", "xml.streaming": true, "xsd.mapping_strategy": "prefer-schemalocation", "core.error_limit": 100, "Working-directory": "C:\\CameraGPS\\example1", "args.FILE": ["C:\\CameraGPS\\example1\\1211300badData.LOG.gpx"], "xsd.external_schemas": null, "xml.user_catalog_path": null, "xsd.assessment_mode": "strict", "xsd.import_strategy": "load-preferring-schemalocation"}
	Selected tool: RaptorXML 2013r2.
2187	Step RaptorXML.xsd.validatexml completed with status: 1 more
2187	Executing FlowForce.move with parameters: {"Overwrite": true, "Source": "C:\\CameraGPS\\example1\\1211300badData.LOG", "Working-directory": "C:\\CameraGPS\\example1", "Destination": "C:\\CameraGPS\\example1\\badData"}

We can click the more link for the RaptorXML validation step for details of the validation error:



When MapForce Server encountered an error in the data mapping step, it terminated immediately and left the XML output file incomplete, and therefore not valid.

We could define job steps to delete the invalid data, but remember, these partial output files can be helpful for diagnosing errors, as we described in the Altova Blog in a post titled Expect the Unexpected – Altova MissionKit Solves a Number Format Mystery.

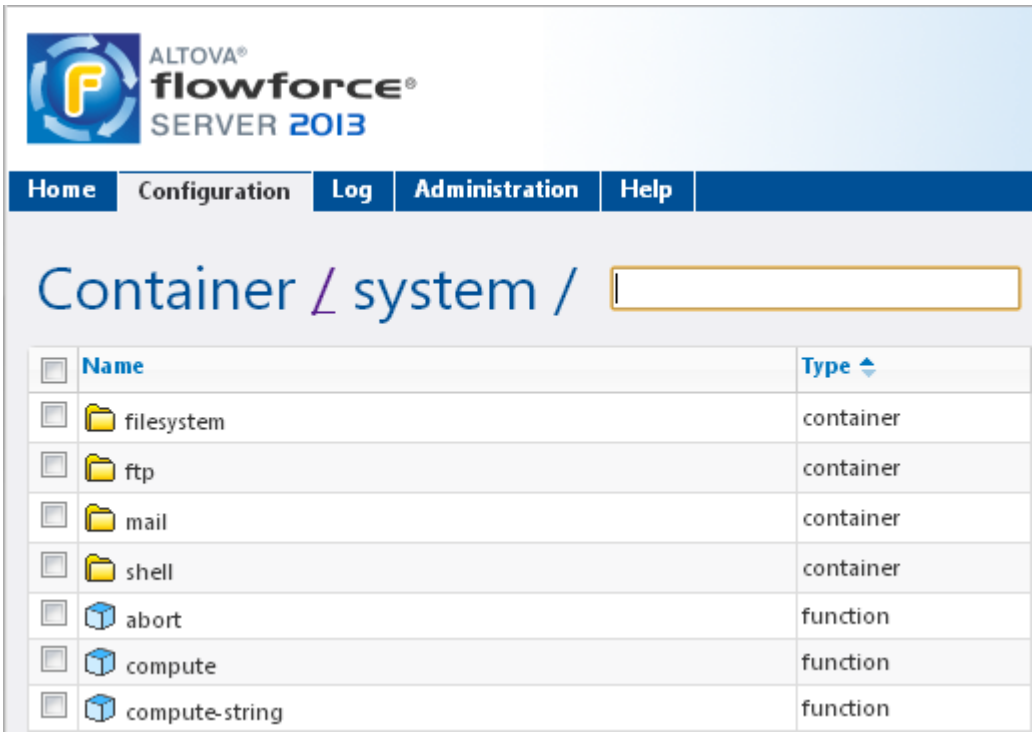
The next chapter covers all the built-in filesystem and other functions you can combine with the job steps we've described so far to make your data transformation, report generation, and XML and XBRL processing workflows efficient and productive.

Chapter 10

Filesystem Commands and More Wizardry with Built-in Functions

Execute command lines or batch files, even to harness external resources

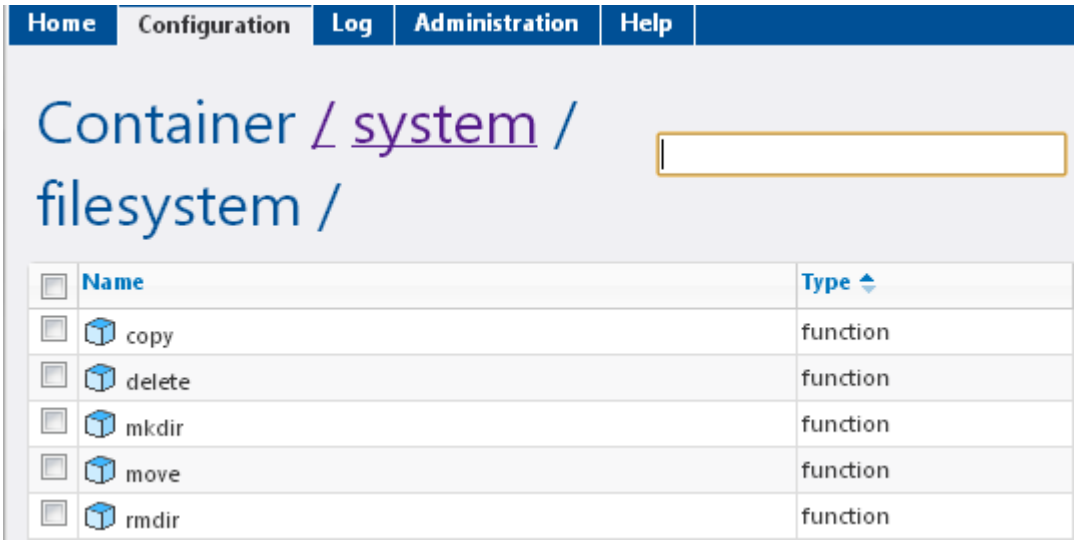
So far we have described jobs that execute MapForce Server for data transformations, StyleVision Server for report and document rendering, and RaptorXML Server for XML processing. In each scenario we also used built-in system functions to copy move, or delete files, as is often required in real-world workflows.



In this chapter we'll take a look at more of the built-in functions automatically installed in the FlowForce Server system container. You can use these commands as execution steps to automate the file housekeeping so often required in enterprise production.

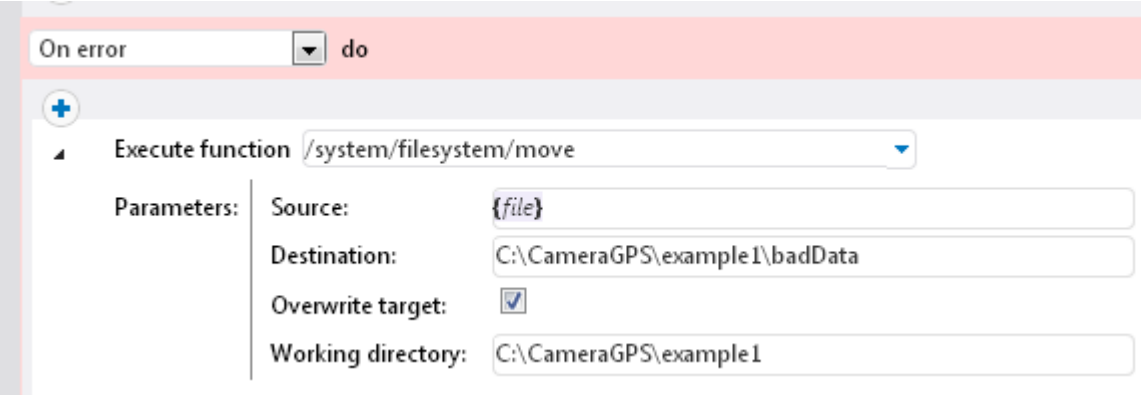
Filesystem Functions

The filesystem container includes the copy, move, and delete functions we used in our job examples, plus mkdir and rmdir to create and delete directories.



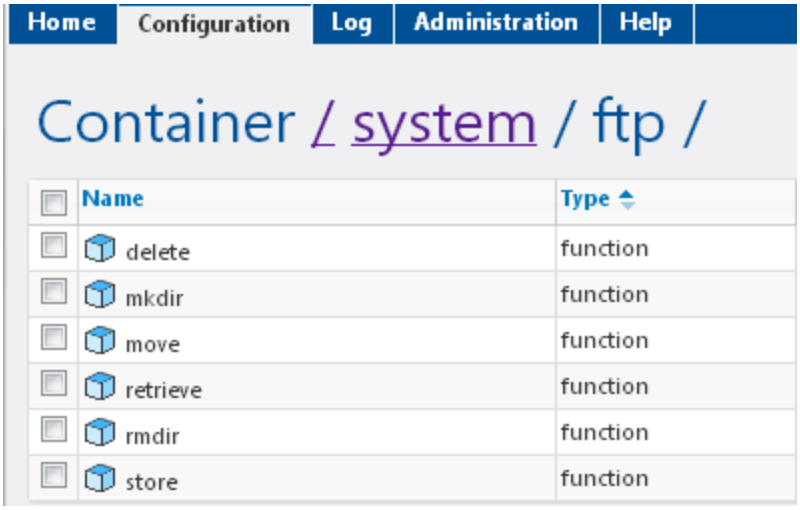
Each function corresponds to a filesystem command, and all parameters required to execute the command are defined in the job.

In our Camera GPS job we created an error path to handle bad input data. We moved input files that failed to a different folder than files that processed successfully. The move job step is shown below, where the file name, destination, and overwrite permission are all defined in the FlowForce Server job itself.

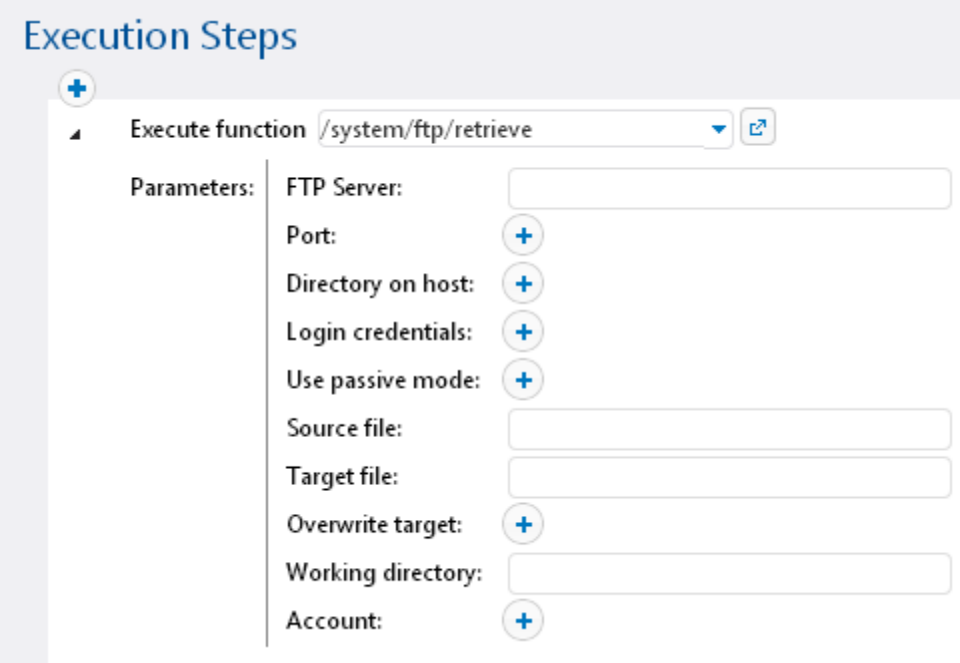


FTP Functions

The FTP container has functions that correspond to commands defined in the File Transfer Protocol (FTP). This means FlowForce Server can interact with FTP servers to transfer files in either direction. Files can be retrieved to become input for a MapForce Server data mapping, StyleVision Server transformation, or for RaptorXML processing, and output files can be delivered to FTP repositories.



As with filesystem functions, all required FTP parameters for a successful transfer are defined in the job. When a user selects an FTP function for a new job step, as shown in the retrieve example below, FlowForce Server automatically provides fields for the appropriate parameters.



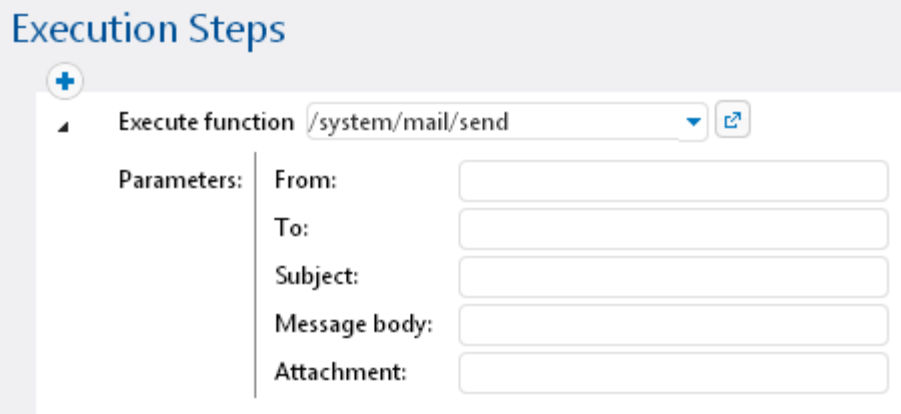
Note that the parameter for the FTP login credential is the login for the remote FTP server, not the FlowForce Server job execution credential. The FTP login credential can be defined locally within the job or stored with other FlowForce Server credentials for shared use

Mail Functions

The mail send function requires configuration of a mail server in the FlowForce Server Administration / Settings dialog. Once that step is complete, a FlowForce Server job can send email messages with file attachments.

Users can define alert emails as part of a job error path, or deliver a report created by StyleVision Server to its intended recipients as an email attachment.

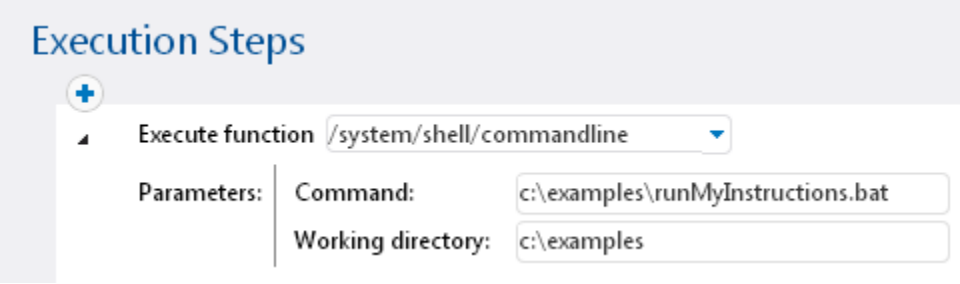
FlowForce Server email is outgoing only. FlowForce Server will not react to incoming email. You can either share an email address monitored by a live recipient, or include a Do Not Reply alert in the message title or body.



Shell Functions

The shell container creates an extremely powerful “do anything” capability. The commandline function allows you to execute an operating system shell command line, which can be a single command, a batch file, or some other executable file.

This means you have nearly endless possibilities to extend FlowForce Server as a scheduling and automation tool for apps and utilities you already developed in-house, or even for other third-party tools.

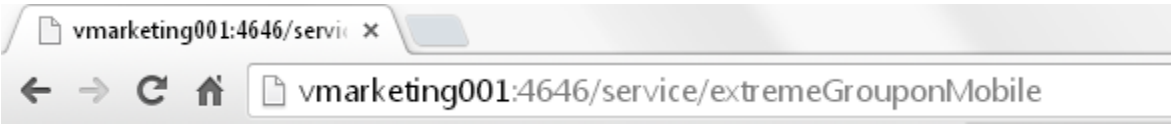


Chapter 11

FlowForce Server Jobs as HTTP Services

Empower end users to execute jobs on demand

FlowForce Server administrators can define jobs as HTTP services to empower end users to execute the job on demand, as easily as opening a Web page.

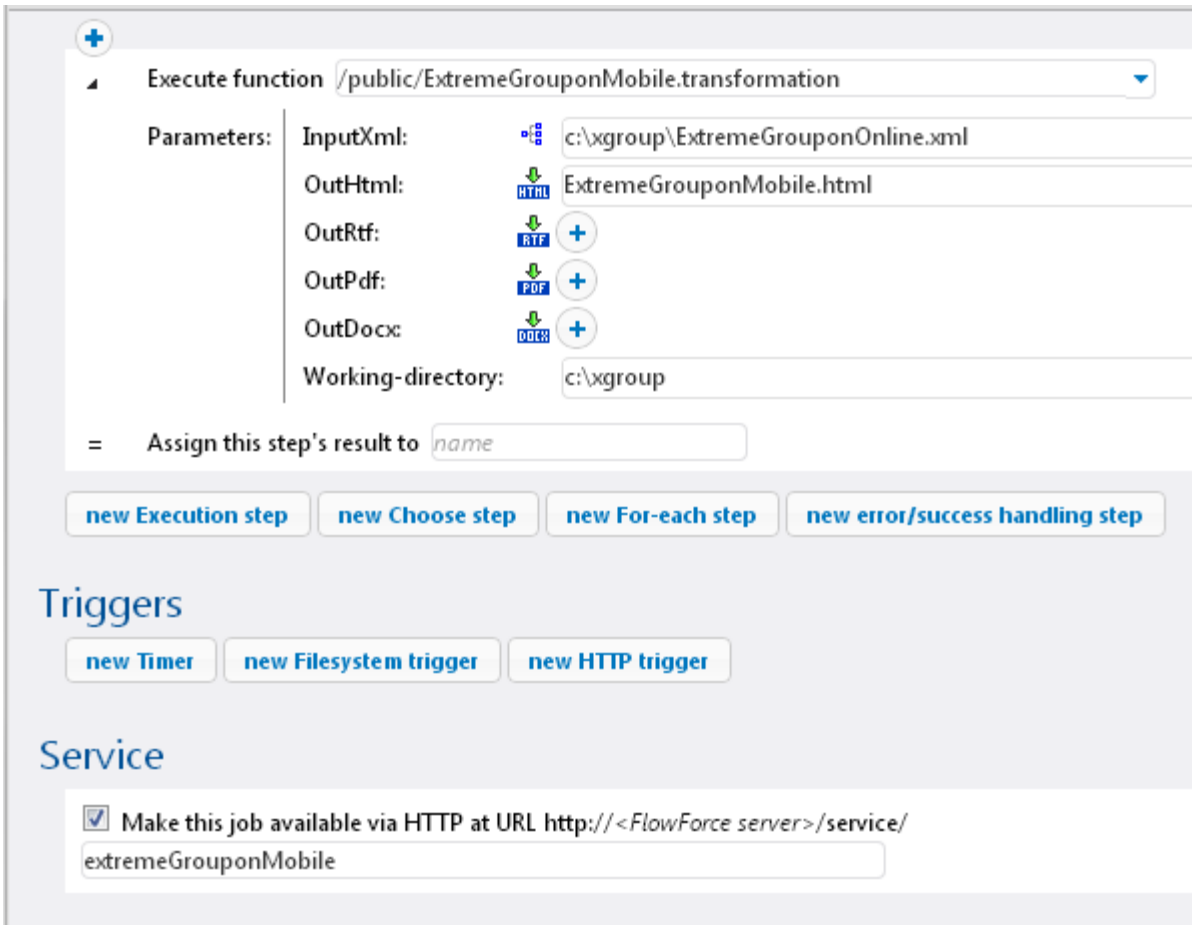


When a FlowForce Server job runs as an HTTP service results are delivered back to the Web browser. If the last step is a StyleVision Server transformation, the job can create a rich HTML-based Web page.

Even better, the same result is simultaneously saved in the enterprise workflow.

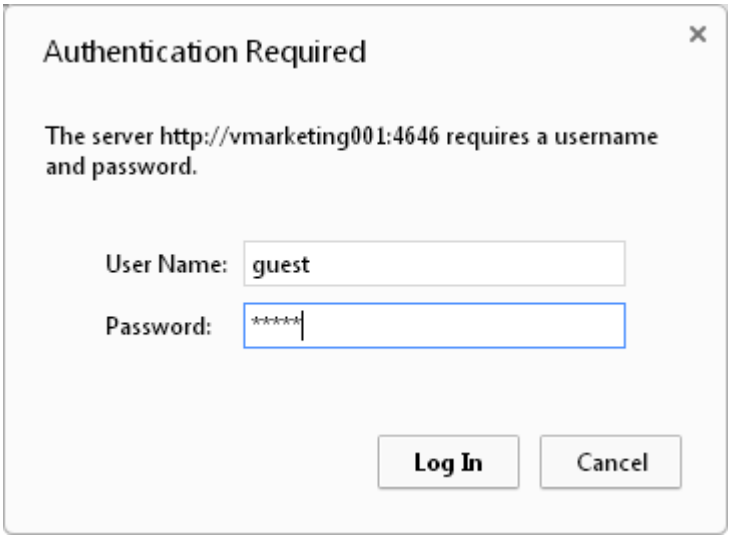
The image below shows the last execution step of a job that queries the Groupon API (we've written about the Groupon API in the Altova Blog, click here for more info). This step defines a StyleVision Server transformation that creates the file ExtremeGrouponMobile.html and saves it in the working directory.

This particular job has no time triggers, file system triggers, or remote server triggers, although any of those triggers could also be applied. Instead, the job is available on demand at the URL defined under the Service heading at the bottom of the image.



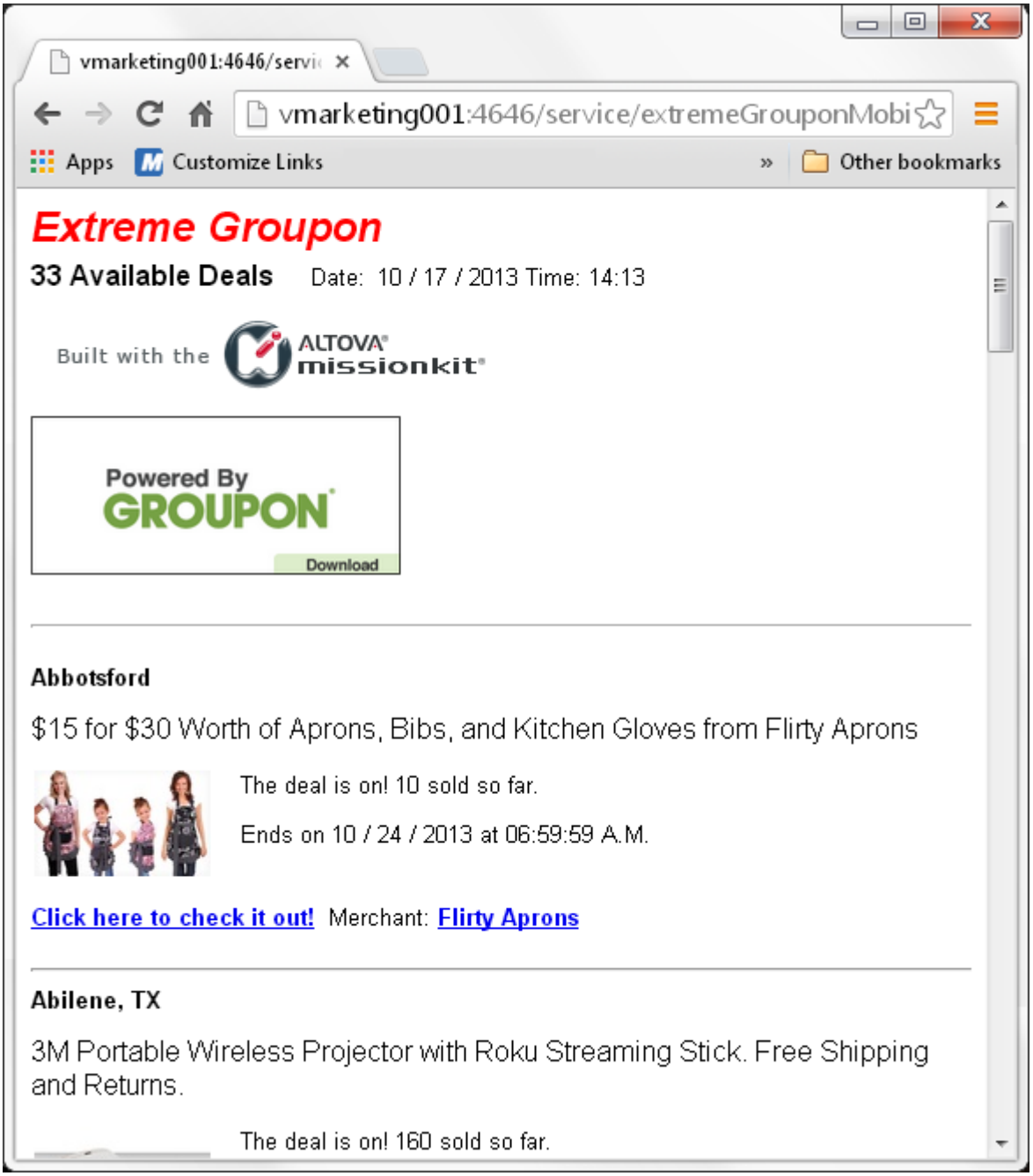
In our example, FlowForce Server is running on a workstation named vmarketing001 with 4646 configured as the port for services, so the complete URL is: http://vmarketing001:4646/service/extremeGrouponMobile

When a user enters the URL in a browser window the first response will be a FlowForce Server login request.



Yes! The service is only available to authenticated users with privileges to access the folder where the FlowForce Server job is stored. FlowForce Server administrators create users and groups and define their access privileges. This allows administrators to define a job for the Human Resources department that cannot be executed by Sales or Purchasing. In our example, a remote user who logs in as guest is only permitted to access jobs in the public folder.

Once the user is authenticated, the job runs and results are delivered to the browser window.

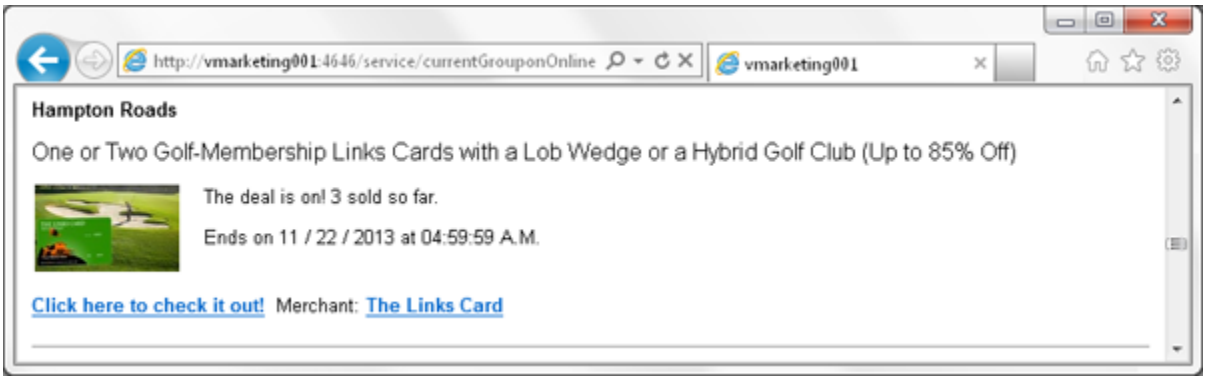

vmarketing001:4646/servi
vmarketing001:4646/service/extremeGrouponMobi
Apps
Customize Links
Other bookmarks
Extreme Groupon
33 Available Deals
Date: 10 / 17 / 2013 Time: 14:13
Built with the
ALTOVA
missionkit
Powered By
GROUPON
Download
Abbotsford
$15 for $30 Worth of Aprons, Bibs, and Kitchen Gloves from Flirty Aprons
The deal is on! 10 sold so far.
Ends on 10 / 24 / 2013 at 06:59:59 A.M.
Click here to check it out!
Merchant:
Flirty Aprons
Abilene, TX
3M Portable Wireless Projector with Roku Streaming Stick. Free Shipping and Returns.
The deal is on! 160 sold so far.

Chapter 12

Result Caching Accelerates Application Response

Satisfy demanding users with instant results

Rapid response to user input is critical to the success of any Web application. FlowForce Server administrators can leverage result caching to deliver nearly instantaneous results to users running FlowForce Server jobs in a browser window as HTTP services.



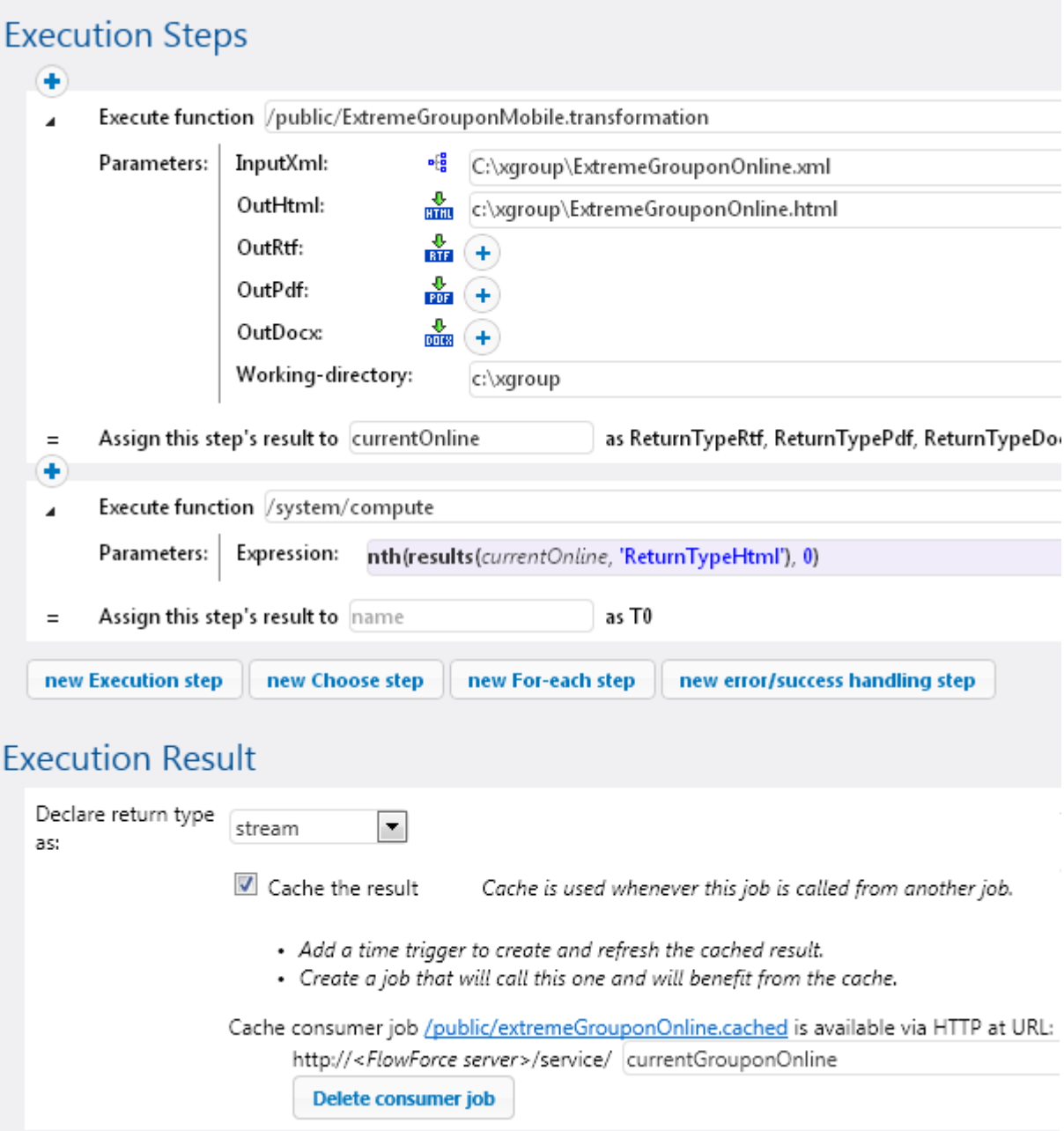
Configuring a FlowForce Server job to take advantage of result caching is a simple two-step process. As an example, let's look at the job we created in the chapter titled FlowForce Server Jobs as HTTP Services. We defined a StyleVision Server transformation as the last step in a FlowForce Server job and made the job available as an HTTP service that delivered the HTML result to a Web browser window.

The drawback to that method is the end user's request only triggers the job to start. Every execution step must be performed successfully before an HTML document is

returned for display in the browser. If the job contains numerous steps for database queries, data transformations, XBRL validation, or other complex operations, or if the server is extremely busy, the end user experiences an unacceptable delay.

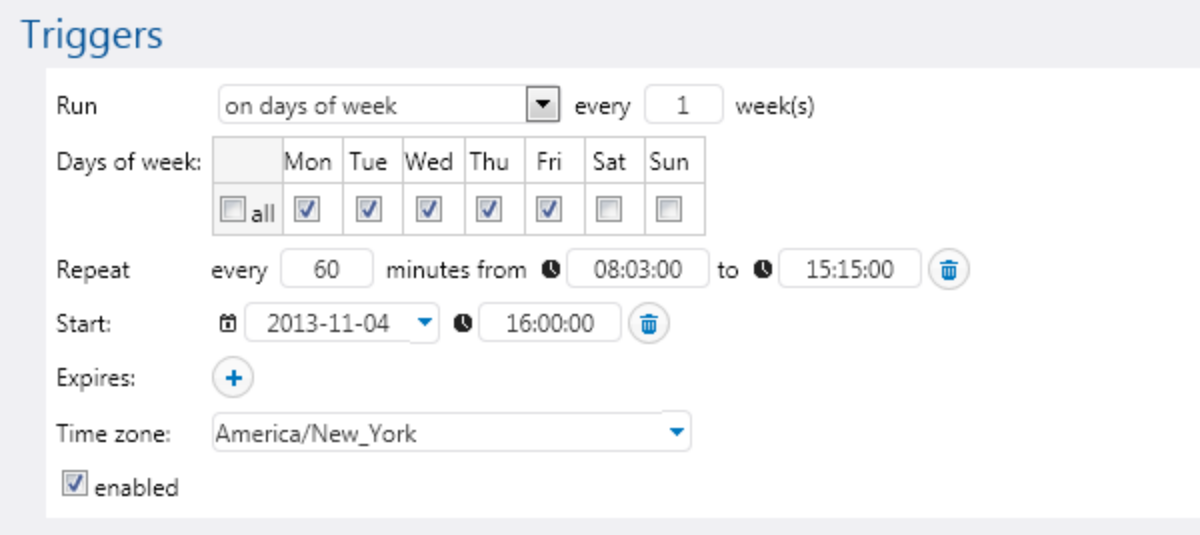
To take advantage of result caching, we will re-define the job to preserve results, set triggers to run on a fixed schedule, and deliver the most recent result on demand to any end user.

The job definition page with caching features is shown here:



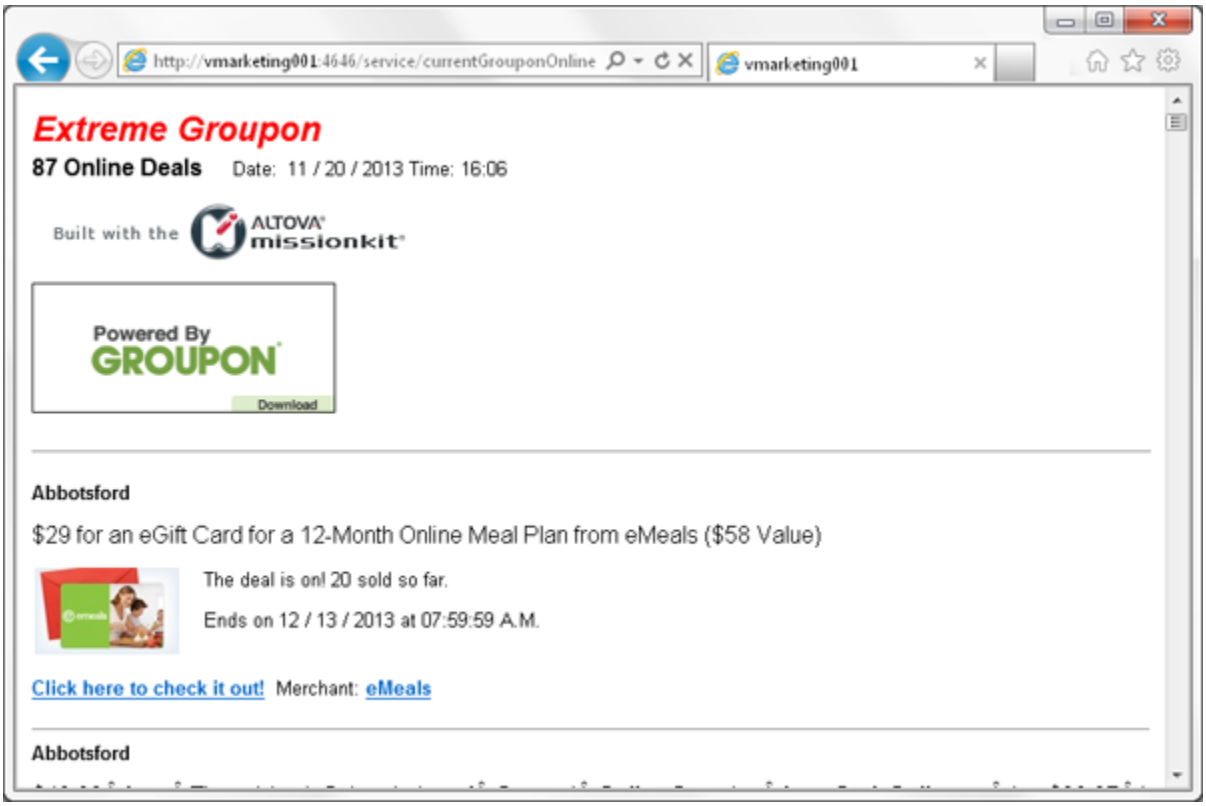
The HTML result of the transformation is assigned to the name currentOnline, and a new execution step uses the /system/compute function with an expression that prepares data for the cache. The Execution Result portion defines the cache and the URL to access the cached data.

We have to run the job at least once to create the cache, so the second step is to define time triggers:



The triggers shown here will execute the job every 60 minutes during business hours from Monday through Friday.

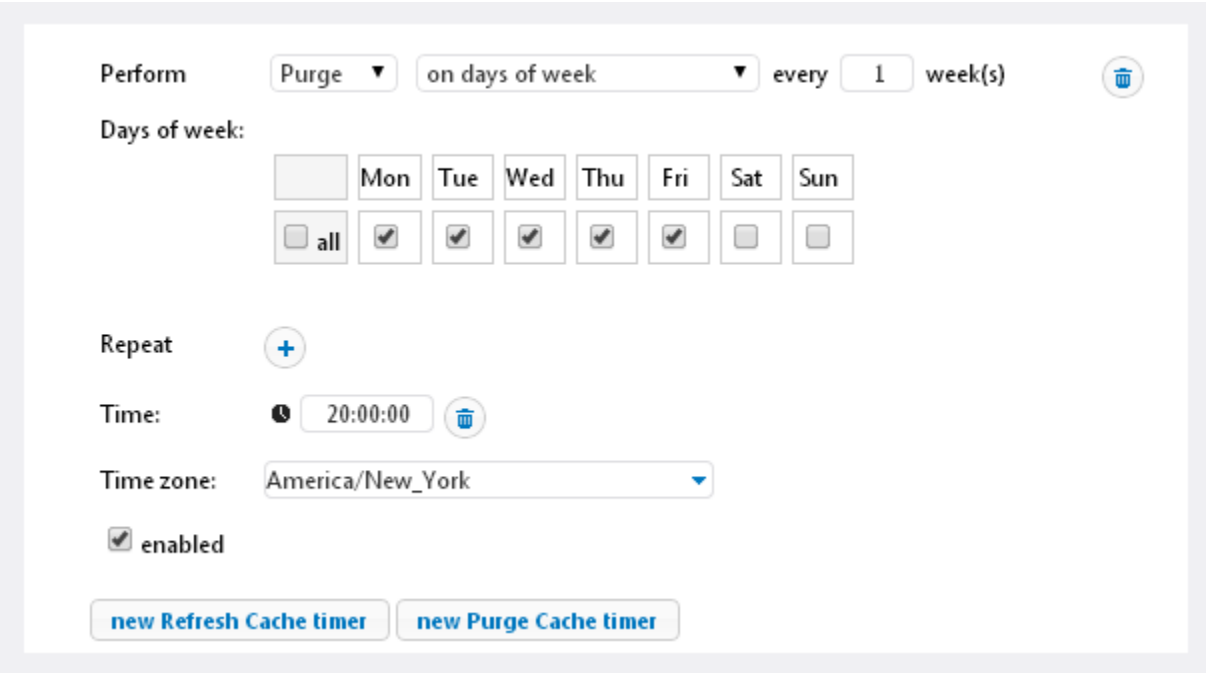
For a FlowForce Server running at vmarketing001, with port 4646 assigned for HTTP services, the URL is http://vmarketing001:4646/service/currentGrouponOnline as shown in the image below:



The entire cached result snaps into the browser window with no processing delay, and instant gratification puts a smile on the end user's face.

Result caching is a good solution to optimize application response time for any job that looks up data from an external source, but is not so time sensitive that it requires last-minute data. One example would be sales detail reports by district and product line for the previous day. Caching is especially beneficial for FlowForce Server enterprise-level data transformation jobs that work with large amounts of data, require complex database queries, or consume Web services where the performance of an external system may not be predictable.

Cache Purge triggers and Cache Refresh triggers can be defined to limit access to stale data. Let's say we want to remove the cache to prevent any further delivery of outdated data every weekday after the close of business hours on the west coast. The dialog to add a new Purge Cache timer looks like this:



The Refresh Cache dialog works the same way. The Repeat option provides even finer control over extremely time-sensitive data by letting administrators define refresh intervals. For instance, an application that uses weather data might want to update

hourly, while a financial application dependent on currency exchange rates could refresh more frequently.

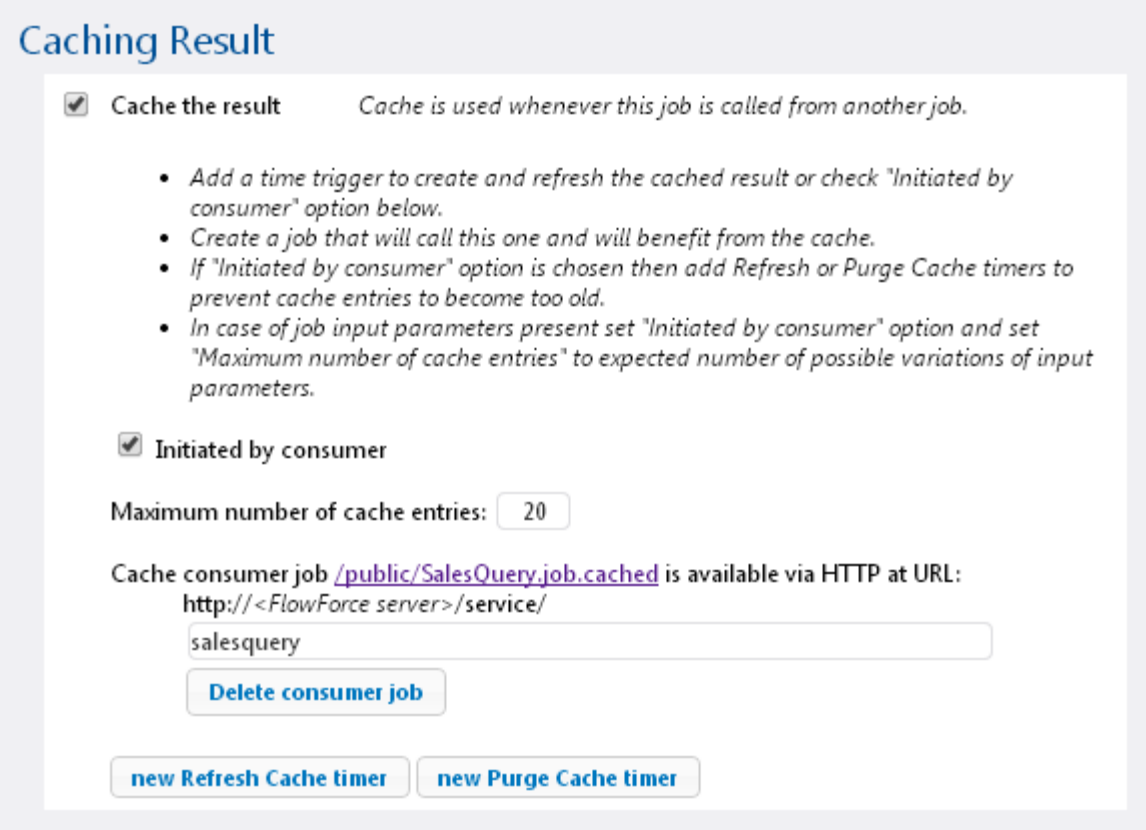
Caching Jobs with Parameters

Result caching is also supported for jobs that use parameters and combinations of multiple parameters. In the job configuration dialog, administrators specify multiple cache entries to match the number of possible parameter combinations that are expected in typical day-to-day usage.

For example, let's say we have a job called SalesQuery that uses parameters to generate unique sales reports for individual regions and products.

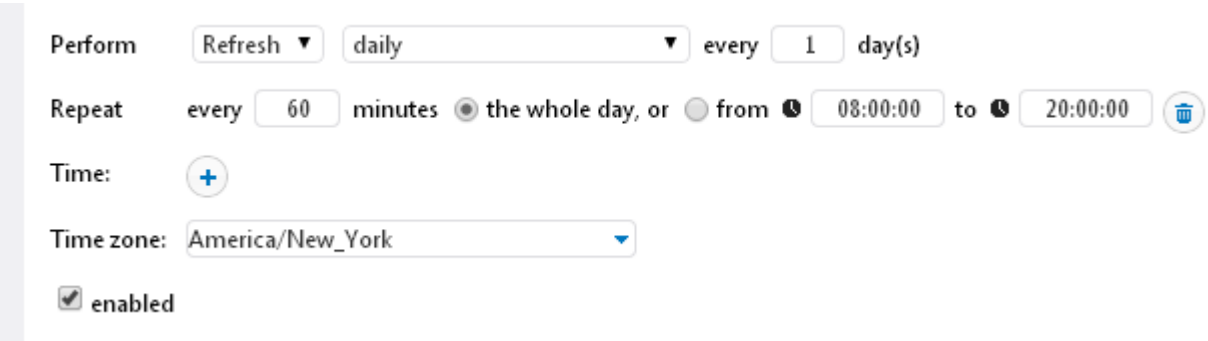
We can define the job as a service that is called via a URL that supplies runtime parameters to select the region and product, such as http://flowforce:4646/service/salesquery?region=East&product=widgets

If there are four sales regions and five products, then a total of 20 unique combinations of parameters are possible. When we enable job caching, we simply define the maximum number of cache entries to match:



The first time the job is run, FlowForce Server records the parameters supplied and caches the result. When the job is run again with the same parameters, the cache entry is instantly supplied.

We can also create a Refresh Cache timer to automatically update the cache:



Most systems define a cache time limit to prevent delivery of stale data and FlowForce Server supports such simple cache expiration limits, too. However, to provide the best possible performance optimization of your data integration projects, FlowForce Server goes much further. The Refresh Cache timer triggers FlowForce Server to automatically

run the job again in the background, using the same parameters, to update the cache. Instead of the typical stale cache expiry, you get automatically refreshed cache entry and can fine-tune the exact performance load on your back-end systems.

And, the same behavior applies for each possible combination of parameters, corresponding to each individual cache entry. Each parameter combination is seeded the very first time it runs, then, based on the Refresh timer, it is continuously updated.

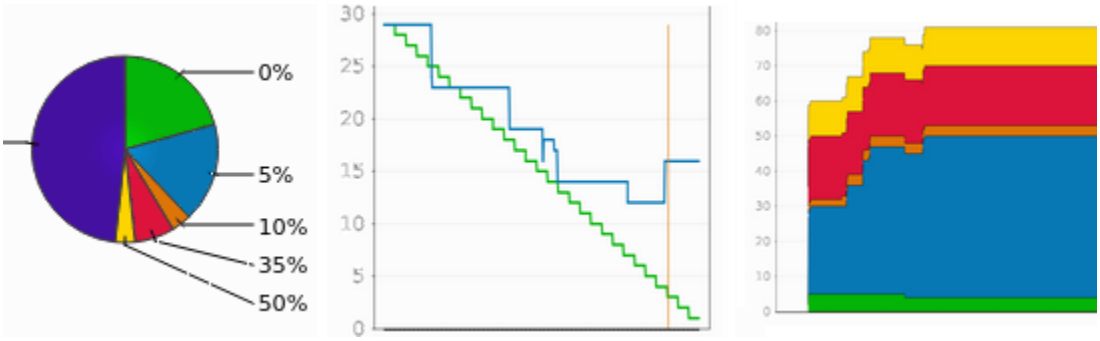
Chapter 13

Installing FlowForce Server in the Cloud

Get powerful data integration functionality without the cost of installing and maintaining another hardware platform

Cloud-first is becoming the new normal. At recent events we have frequently been asked about using FlowForce Server in the cloud. The answer is definitely, go for it. The installation is easy.

In fact, we use Altova Server Software products ourselves for an internal reporting application, installed on local virtualized servers and on an AWS cloud instance. The charts below were generated by StyleVision Server running in the cloud to quickly communicate information about changes in dynamic data.



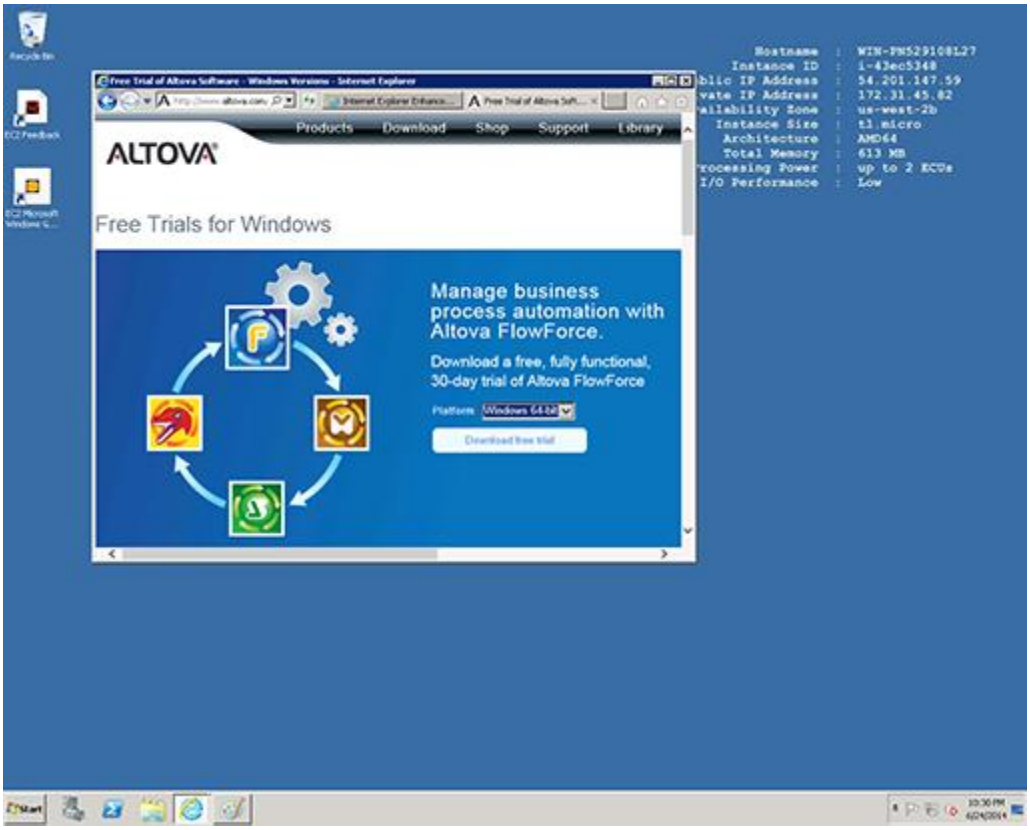
StyleVision Server is based on the built-in report and document generation engine developed for StyleVision and renders .SPS stylesheets originally designed in StyleVision, including features like a rich variety of charts to visually represent data.

In this chapter we will walk through the installation of FlowForce Server, MapForce Server, StyleVision Server, and RaptorXML Server for a complete data integration solution in the cloud.

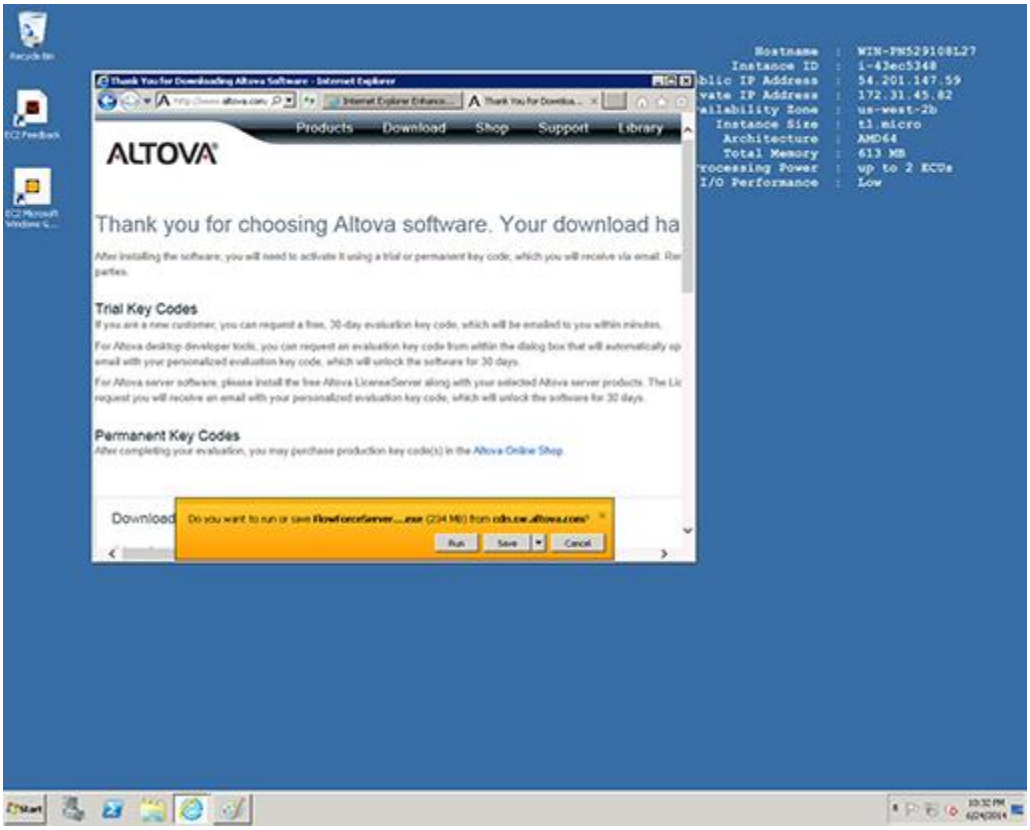
Most cloud providers offer a variety of preconfigured virtual machines. All you have to do is select a virtual machine that matches one of the Windows or Linux operating systems supported by Altova Server tools. Follow the cloud vendor's instructions to permit access to your virtual server in the cloud from IP addresses of your local network.

By the way, the same process we describe below works equally well for a VM instance running inside your own local network.

The screenshot below shows a Windows Server 2008 R2 SP1 instance hosted by Amazon Web Services. You can do the same thing on Windows Azure, Rackspace, or any other leading cloud provider. We connected to our cloud server via Remote Desktop, launched Internet Explorer, and navigated to the Altova Web page for Server Software downloads.

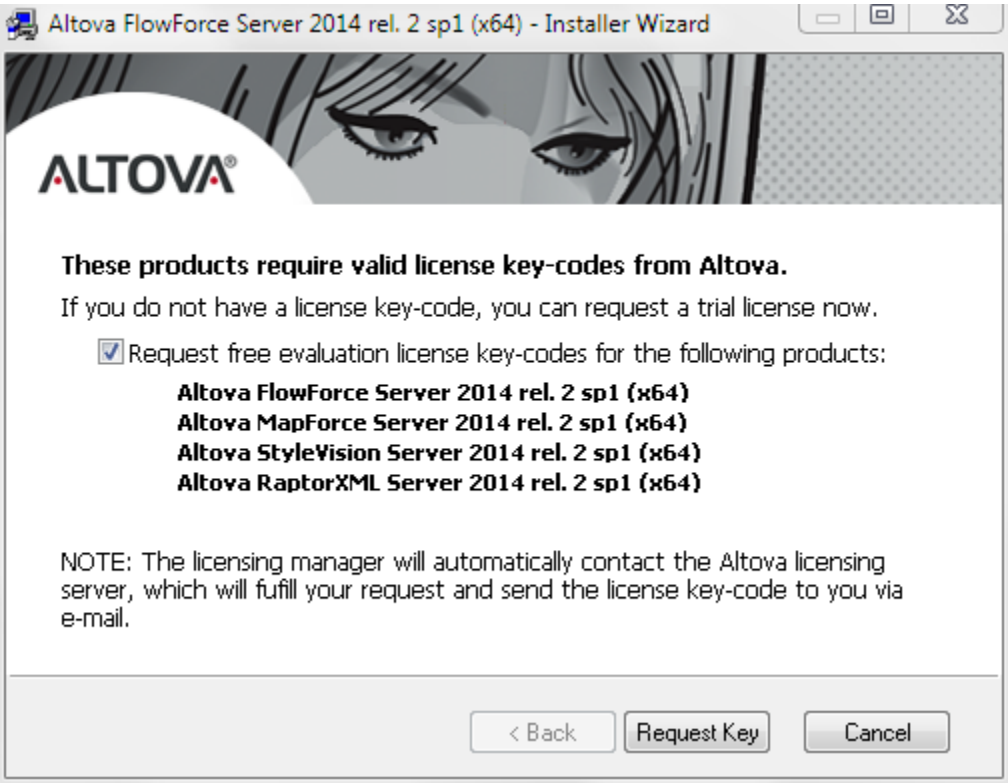


When we download the installer, IE asks whether to run or save the file. To save time, we can go ahead and run it.



When you run the FlowForce Server installer, you can also choose to install MapForce Server, StyleVision Server, and RaptorXML Server at the same time. These additional tools are all executed automatically by FlowForce Server when needed to complete multi-step data integration requirements.

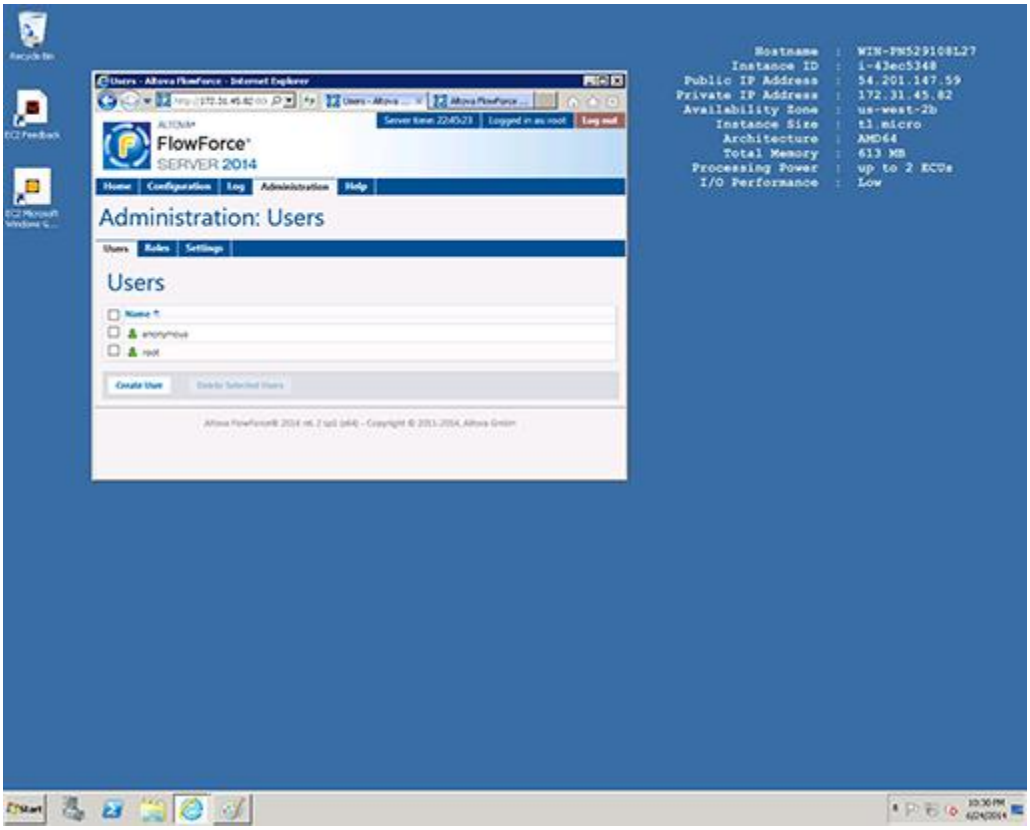
The installer can even request evaluation license key-codes for a fully-functional trial:



Altova Server Software is licensed based on CPU cores. The number of cores licensed must be greater than or equal to the number of cores available on the server, whether it's a physical or virtual machine. For example, if a server has 8 cores, you must have at least an 8-core license.

If you let the installer generate a trial request, the evaluation key-code will be automatically matched to the configuration of your cloud server instance.

And here is the payoff: FlowForce Server running in the cloud. Shown below is the user setup window where an administrator can create users and define their privileges.



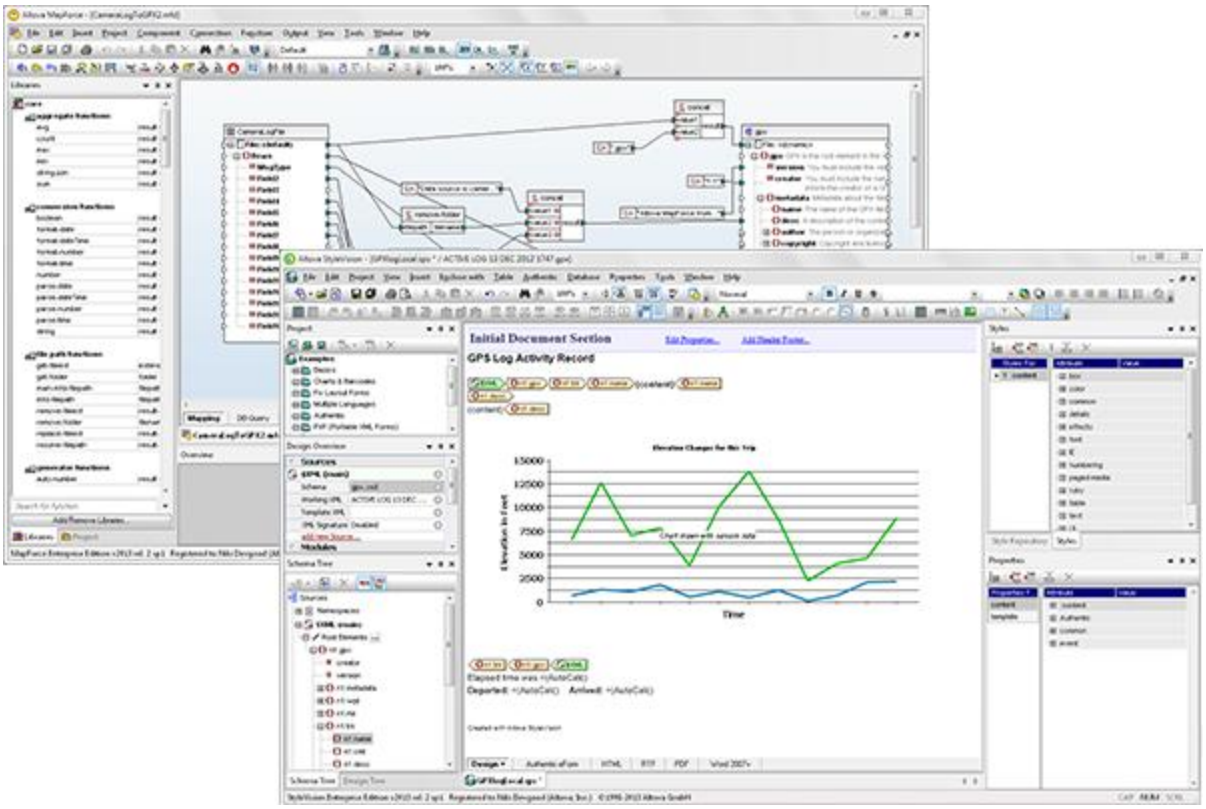
The Help button in the FlowForce Server Web interface opens the complete integrated manual in a new browser tab. Or, you can download the manual from the documentation page in the Support section of the Altova Web site to review up front.

Chapter 14

Download Example Files for FlowForce Server, MapForce, and StyleVision

Recreate the examples from previous chapters and more

In response to an interested reader's suggestion, the Altova MapForce mapping files and StyleVision stylesheet we deployed to FlowForce Server for the job described an earlier blog post and in the chapter Taming Bad Input Data with FlowForce Server are now available for download on the Altova Web site at www.altova.com/documents/AltovaBlogExampleFiles.zip



Simply unzip the archive into a new folder and you'll have all the data mappings, the stylesheet, and other supporting data files all in one place. A ReadMe file explains the contents. You can download fully functional free trials of Altova MissionKit and FlowForce Server at http://www.altova.com/download.html and implement and test FlowForce Server yourself.

Or, execute the data mappings in MapForce and the stylesheet in StyleVision to see how easy it is to extract meaningful information from the GPS data recorded by your own digital camera or GPS device. The example files in the download were also used in posts to the Altova Blog titled Web Service as a Look-Up Table to Refine GPS Data, XPath Enhances XML Reports, and others in our series on working with XML and Global Positioning Systems.

If you're already an Altova MissionKit user, you can download these files with examples of Web services and user functions for MapForce, and XPath calculations and chart features for StyleVision, and add them to the extensive libraries of MapForce and StyleVision samples installed with Altova MissionKit tools.

Afterword

Thank you for reading this book. You can keep up with the latest updates to FlowForce Server and other Altova products by reading the Altova Blog.

And get even more details on the Altova Web site or even download the complete FlowForce Server User and Reference Guide!

"We evaluated Altova MapForce against all the major data integration applications in the industry and found it to be the most powerful and easiest to use by far."

Mark Beede
Senior J2EE Consultant for HealthTrans