



EnRoute

EnRoute 3.0
February 15, 2016

InPress
Systems 

Copyright © 2016 InPress Systems AB.

This manual and the software described in it are provided under licence and may only be used subject to the terms of this licence. Any unauthorised duplication or distribution is prohibited by law. The content of this manual is provided for information only, is subject to change at any time and should not be construed as a commitment by InPress Systems AB or its partner companies. InPress Systems AB accept no responsibility for error or omissions that may appear in this document. Unless specifically authorised in the licence agreement, no part of this document may be reproduced, stored in a retrieval system or transmitted in any form without the prior written consent of InPress Systems AB.

Xinet, FullPress, WebNative and WebNative Portal are trademarks of North Plains.
Archiware and P5 are trademarks of Archiware GmbH
Amazon Glacier is a trademark of Amazon Inc.
All other trademarks are the property of their respective owners.

CONTENTS

1	WHAT IS ENROUTE	4
2	INSTALLING ENROUTE.....	5
	2.1 Platform requirements.....	5
	2.2 Where do I get the software.....	5
	2.3 Installing the software	5
3	CONFIGURING ENROUTE.....	6
	3.1 Base configurations	6
	3.2 Event list.....	7
	3.2.1 Event schedule indicators.....	8
	3.2.2 Creating a new Event.....	8
	3.2.3 Pausing events.....	9
	3.2.4 Editing and adding schedules.....	9
	3.2.5 Editing and adding conversion setups.....	10
4	ENROUTE EVENTS.....	12
	4.1 General event Setup	12
	4.2 Scan Setup	13
	4.2.1 Scan, Scan files, Scan folders & Scan recursively.....	13
	4.2.2 Search database using Keywords	14
	4.2.3 Search database using Template.....	16
	4.2.4 Scan Database content in named folders.....	17
	4.2.5 Scan uploadreports	18
	4.2.6 Scan XML files	18
	4.2.7 Fetch pop mail	21
	4.2.8 Filelist limitations.....	21
	4.2.9 Using database field for notifications	21
	4.3 Routing Method	22
	4.3.1 Move files.....	22
	4.3.2 Convert files	23
	4.3.3 Transfer files via ftp/sftp.....	24
	4.3.4 Send link to asset download.....	26
	4.3.5 Run Custom Program	28
	4.3.6 No routing	29
	4.3.7 Routing Paths and Custom program arguments	29
	4.3.8 Updating Database Fields.....	33
	4.4 Reporting & notification	37
	4.4.1 Email notifications.....	38
	4.4.2 Email notification template format	38
	4.4.3 Email notification data and tokens.....	41
	4.4.4 Reporting	47
	4.4.5 Report template format.....	47
	4.4.6 Report data and tokens	50
	4.4.7 Report examples.....	50
	4.4.8 Connect to external mySQL	52
	4.5 Testing and Running events from admin.....	53
5	ACTIVITY AND LOGS	54
A	TIME AND DATE FORMATTING.....	56
	INPRESS SYSTEMS SOFTWARE LICENSE AND CREDITS	57

1 WHAT IS ENROUTE

EnRoute is designed to be run at the central server where all assets are stored. It has a back-end engine and a web-based administrators interface. The **EnRoute daemon, enr**d, runs in the background and is started automatically when the server starts up. The daemon is responsible for running the events scheduled by the administrator (see below).

EnRoute allows an administrator to configure **events to be run** with a **cycle time** or using one or more **schedules**. A cycle can be setup with a minimum time (Hot folder) or a specific interval (minutes or hours). A schedule can be used to run an event at specific hours on certain weekdays or dates. It is possible to select multiple schedules but only one cycle. It is not possible to combine a cycle with a schedule.

An event executes the operations of **scanning**, **routing** and **reporting** in that order.

The **scanning stage** finds the assets to operate on based on the event setup. Typical scanning setups can be: *Scan directory for files* or *Search directory using keywords*. The scanning process takes into consideration locking of files, growing files, etc. Only files that are stable will be included in the scanned set. Other scanning options are available to, for example, ensure that the asset has generated a proper entry in the Xinet database before being included in the scan result set.

When assets have been found, a **routing method** is applied. Typical routing setups can be: Move assets to path *defined by certain database fields* that have been applied to the asset, Move assets to path *defined by data supplied by the user* using a sidcar XML file. Routing can also set Finder flags (colors), apply keywords to the routed files and run conversion on assets. It is also possible to use the route step to generate an archive request in InPress OnFile or to pass the found assets to a custom program.

The last step, **reporting**, makes it possible to send *notifications by e-mail* and/or *write textbased reports* of the operations performed. All e-mails and reports can be customized using InPress Systems notification templates, common for most of InPress Systems products (Accelerator, InterAct, InAlias and OnFile).

2 INSTALLING ENROUTE

2.1 Platform requirements

EnRoute is available for MacOSX (PPC/Intel) and RedHat Linux 5+. The RedHat distributions are available in two versions: 5 and 6-64.

The version 5 release is a 32 bit release. When installed on a RedHat 5 64-bit linux systems it requires the compatibility packages to be installed. This release can only be used on linux v 5.

The RedHat 6 version is 64-bit only and cannot be used on a 32 bit system. It can also be used on Linux version 7.

EnRoute version 3 requires Xinet 17.X or later.

IPIK (InPress Portal Integration kit) is required when using the Assetlink routing method in conjunction with an outward facing Portal server. IPIK needs to be installed on the Portal server. No configuration of IPIK is required for EnRoute assetlink.

2.2 Where do I get the software

To get the latest EnRoute software, use the downloads page at www.inpress.se.

The installation package is a gzipped unix tar file named “enr” followed by the version and platform and ending in “tar.gz”. For example enr.3.0.redhat6_64.tar.gz refers to EnRoute version 3.0 for the RedHat 6 operating system.

2.3 Installing the software

Login to the server as root and place the software in an appropriate installation directory. Uncompress and untar the file to display the “enrouteinst” folder, a README file and an installation script.

Run the script by typing “./installenroute” and press return. Any current installation will be upgraded to the version you are installing.

2.4 Starting up

The installation script installs the proper startup scripts for all platforms and then starts the daemon. The daemon should start and stop using the normal startup procedures.

In case you need to stop the daemon manually, run the following command line command

```
# /usr/inpress/enroute/bin/enroute_ctrl stop
```

In case you need to start the daemon manually, run the following command line command

```
# /usr/inpress/enroute/bin/enroute_ctrl start
```

2.5 Upgrading from version 2.X

EnRoute version 3 is installed in a different location than earlier versions. The configuration is also kept in a database as opposed to configuration files. The existing version is automatically imported into the new when installing. The old folder structure is renamed and not removed. The old daemon is deactivated.

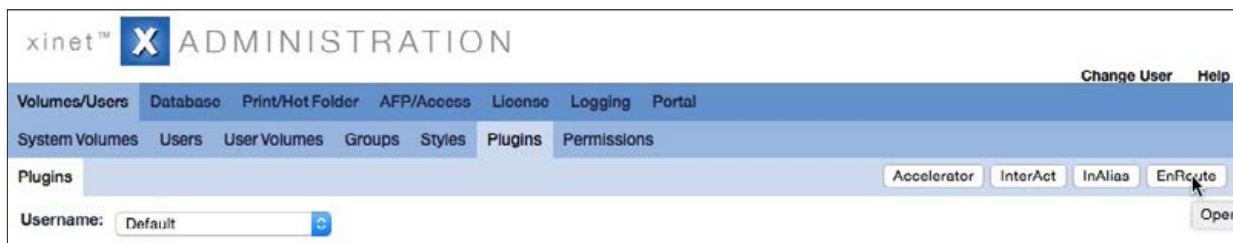


You need to be logged in as root when installing the software

3 CONFIGURING ENROUTE

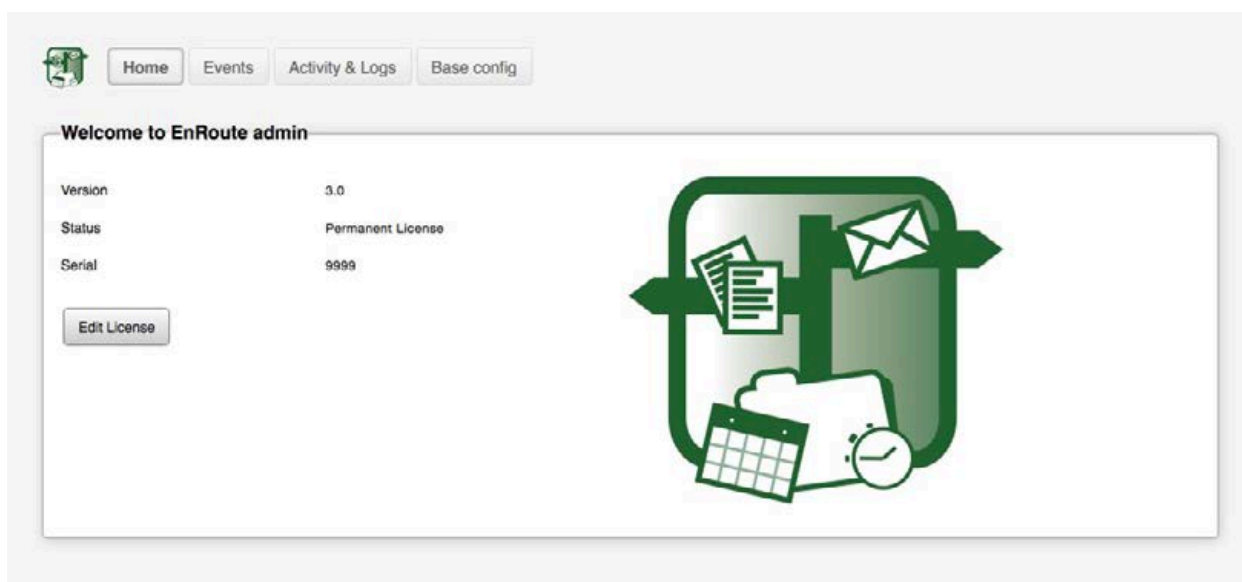
The web-based EnRoute admin utility can be found under the web address or via the "EnRoute" button under the Xinet nativeadmin plugins tab.

<http://<yourserver>/webnative/plugins/enrouteadmin>



Access EnRoute admin from nativeadmin plugin page

You need to be logged in as “nativeadmin” to get access to the admin utility. It is organized the same way as the other admin utilities for InPress Systems, with buttons for different functions and licensing.



EnRoute admin GUI

3.1 Base configurations

There are some base configurations that can be done in EnRoute. These configurations will be default for all the events. Some of the configurations can be overridden by the event configuration.

CONFIGURATION	DESCRIPTION
Sleep time	How long the daemon will wait between runs (seconds). The default is 10 s
Maximum concurrent events	The maximum number of EnRoute events that can run simultaneously. The default is 5.
Error E-mail	E-mail address to send error message to. Can be overridden by Event. This global Error email address is also used when the daemon reports an error.
Reply-to E-mail	The sender of all EnRoute email messages. This setting can be overridden per Event.

Main Server URL	Server url that will be used when building links to WebNative for notifications and reports and for certain links in event emails (like Asset link). This should be the address to the Xinet back end server as seen from the Portal server in the case of one being used and configured below. This setting can be overridden per Event.
Portal Server URL	Server url that will be used when building links to WebNative Portal for notifications and reports and for certain links in event emails (like Asset link). This setting can be overridden per Event.
Portal Site name	Portal site name that will be used when building links to assets in WebNative Portal for notifications and reports. This setting can be overridden per Event.
Display date format	Select the format of dates and times to be used in the administration interface.

Base configurations

3.2 Event list

The central concept in EnRoute is the event. Each event is configured and then scheduled to be executed in a cyclic manner with a cycle time or via one or more schedules which can be created.

The Event section present a list of all the events that have been configured. The list show the name and description as well as the current scheduling information such as selected schedules and cycles, last execution and next execution. The screen also act as the access point for editing the events and viewing event logs. From this page it is also possible to add new events.

Schedules to be used for events as well as conversion setups can be viewed and edited from the Event page.

Details	Next	Last	#	Event description
Archive-ingest Every_friday_at_5	2016-01-29 17:00:00	-	-	Search for jobfolders ready to be archived
Route-Incoming 5 min	-	2016-01-21 16:06:30	7	Routing incoming files on date
Route-to-ftp 1 min	-	-	-	
Searchtest-1 1 min	2016-01-26 12:30:26	2016-01-26 12:29:24	7	
Weekly-clean weekend_morning Every_friday_at_5	2016-01-30 08:00:00	-	-	

Process id for enrd is 8900. Process is Running

Buttons: Add new event, Schedules, Conversions, Pause all, Unpause all, Refresh

The Event list with indicators, information and buttons for edit, pause and view log

3.2.1 Event schedule indicators

The Event schedule indicators consist of a "colored light", a symbol for the type of execution and the details of the selection execution options.

- Green light : At least one execution option is selected and have been enabled
- Yellow light : No execution options have been selected
- Red light : At least one execution option is selected but none have been enabled
- Grey light : The event has been paused

The symbol can either be a calendar or two curved arrows:

- Calendar : The execution option or options are of the **schedule** type
- Arrows : A **cyclic** execution option have been selected

The next and last indicators show when the event was executed the last time and the time it is due for execution again.

3.2.2 Creating a new Event

To create a new event, use the "Add new event" button. A dialogue where the name of the event is defined is shown. The name can only consist of the letters a-z, A-Z, numbers and hyphen (-), underscore (_) and dot (.). The name can not be changed afterwards. When adding a new event configuration it is possible to copy an existing event as a starting point.



Event names can only include letters a-z A-Z, 0-9, "-", ".", and "_"

Add new event

Route-incoming

Copy event

Add Close

Adding a new event

A new event has been added

Edit the event tabs and use the save button. See section 4 for full information on the Event scanning, routing methods and reporting.

3.2.3 Pausing events

An event can be temporarily paused by using the Pause button. The event will then be marked as paused and all execution of the event will be postponed. Note that if the event is running while setting it to paused it will still finish the execution. The event will then be executed when unpaused providing the next execution time has been reached.

It is also possible to pause all events with one click using the Pause all button.

Pausing does not change any settings on the event.

3.2.4 Editing and adding schedules

An event can be executed on a regular interval using a cycle or on certain times using schedules. The schedules are all custom made and consist of what weekdays or dates to run as well as the hour. Several schedules can be created and reused on different events.

Schedules are created using the Schedules button at the bottom of the event list page or the button next to the execution options on the event configuration page. The name can only consist of the letters a-z, A-Z, numbers and hyphen (-), underscore (_) and dot (.). The name can not be changed afterwards.



Schedule names can only include letters a-z A-Z, 0-9, "-", "_", "." and " "

Input a name and add a schedule

Edit the schedule by checking days, dates and hours

The list of schedules

3.2.5 Editing and adding conversion setups

Editing and adding conversion setups is similar to schedules. Open the list window from the button at the bottom of the event list page or next to the conversion selector in the event. Note that the button in the event configuration page is only available when the routing method convert has been selected.

The name can only consist of the letters a-z, A-Z, numbers and hyphen (-), underscore (_) and dot (.). The name can not be changed afterwards.



Conversion setup names can only include letters a-z A-Z, 0-9, “.” “ ” and “ ”

Input a name and add a conversion setup

Edit conversion : Webstore

File Format	JPEG Medium
icc profile	sRGB
Color space	RGB
Resolution (dpi)	72
Scale method	Pixel
Width	250
Height	250
Padding	Yes

Edit the conversion setup

Routing Method

Routing Method ?	Convert files
Delete original files ?	No
Unzip ?	No
Save original zipfile ?	/path/where/to/route/the/original/zip

Route path

Conversion option ?	<div> <div>Thumbnail</div> <div>Webstore</div> </div>	<input type="button" value="Conversions"/>
Handling of non-images ?		

Select the conversion setup from the event configuration's routing section

4 ENROUTE EVENTS

An Event consists of three steps: Scanning, routing and notification/reporting.

Scanning decides how EnRoute finds the files to work on. Scanning can be done via searching through folder structures in different ways or by querying the Xinet database to find files.

The *Routing method* decides what to do with the items that has been identified in the scan step. The methods provided are copy/move, convert, send via ftp, send as downloadlink, ingest into OnFile archive (requires an OnFile license) and pass to custom program/scripts.

Notification and Reporting controls how e-mail notifications and text based file reporting should be handled. It is also possible to send information to an external MySQL server or post to another server via an http POST.

4.1 General event Setup

The event's general setups relate to scheduling, description and execution of a second event after the event is done.

OPTION	DESCRIPTION
Event description	A descriptive text that is not used anywhere in the event.
Event id	<p>A unique identifier created when the event runs. Can be referred to as event.ID when routing files, sending emails, writing reports and setting database fields.</p> <p>The Event ID is a string created much like routing paths (see more details in the routing section). The string is build using functions and tokens concatenated with "&", Example: "ID_&date(%Y-%m-%m.%H.%M)"</p> <p>Functions that can be used are: date(format) = execution date where format is the standard UNIX date formatter, default is YYYY-MM-DD. counter(N) = event counter where N is the number of significant digits. Using a number for N specifies the counter with leading "0" characters.</p> <p>The default value is a unique numeric identifier.</p>
Run event	<p>Select the cycle or schedule(s) to be used for the event. It is possible to select a single cycle or multiple schedules. It is not possible to mix a cycle and schedule.</p> <p>Each selected cycle/schedule need can be activaded separately, See illustrations below</p>
Execution info	Information about the execution status. The last time when it was executed as well as the expected next time of execution. Next execution will not show if no cycle/schedule is active. Next execution is updated when the event is saved.
Trig another event when done	Select an event to be run when the current event has executed. Note that the event will not be executed if the current events exits because of no files found.

Some examples of making the run event setup follow below:

The screenshot shows the 'Execute event' configuration window. The 'Run event' section has '5 min' selected with a checked box and an 'x' icon. The 'Execution info' section shows 'Last executed at: -, Next execution set to: -'.

A cycle has been selected and activated by checking the box. No more options can be selected.

The screenshot shows the 'Execute event' configuration window. The 'Run event' section has 'Every friday at 5' selected with an unchecked box and an 'x' icon. A dropdown menu is open showing 'Add schedule' (checked), 'weekend morning', and 'Schedules'.

A schedule has been selected. It is possible to select more than one.

Execute event ?	Every_friday_at_5 <input checked="" type="checkbox"/> x	weekend_morning <input type="checkbox"/> x
Execution info ?	Last executed at: -, Next execution set to: 2016-01-22 [Fri] 17:00:00	

Two schedules have been selected and one activated. The event has been saved and next execution reflects the selection.

Execute event ?	Every_friday_at_5 <input type="checkbox"/> x	weekend_morning <input checked="" type="checkbox"/> x
Execution info ?	Last executed at: -, Next execution set to: 2016-01-23 [Sat] 08:00:00	

The next execution time reflects a change has been made after the event has been saved.

4.2 Scan Setup

There are a number of scanning methods: file system scans, database searches, file parsing and email retrieval.

File system scans are performed within a directory selected using the navigation tool. Navigation is provided within all defined Xinet volumes. If the scan need to happen in a non-Xinet enabled folder, just type in the path.

The file system search can be made for **files**, **folders**, **files and folders** or **files recursively**. It is also possible to restrict the search in multiple ways. See details in following sections.

The two file parse scan methods open the files found and parse the content in order to find file information and additional metadata that can be used in the route and notification steps. The **uploadreport** scan read the upload reports generated by the WebNative upload cgi. The **xmlfile** parse read an XML file to get the same information. The XML file needs to conform to a format defined below.

There are two different **database query** scan scan methods. The first one performs a search using keyword data while the second searches for content in a **named parent folder**. Both queries can be restricted in many ways and require a base path where to perform the search. See more examples of queries in following sections.

The no scan option can be used in conjunction with the custom program route method to schedule programs to run.

See details of the different scan types in the following sections. A few options for scanning are common to all the scan methods:

OPTION	DESCRIPTION
Run with no files	Run the setup, even if there are no files found. Routing and reporting will take place.
Scan wait (s)	Enroute uses this wait time to confirm that a file is not growing in size. Default is 5 seconds
Wait for DB	When routing using database fields, it is important that the file has been updated in the database before routing is performed. This setting forces EnRoute to wait until the file has been added to the Xinet database before being added to the active filelist.

4.2.1 Scan, Scan files, Scan folders & Scan recursively

The filesystem scan function in EnRoute come in several flavors.

Scan, **Scan files** and **Scan folders** search the first level of a scan path. It will either scan for files and folders or just files or folders.

The **Scan recursively** option will scan for files only but recursively in a path. Folders will be ignored. When this scan type is used, there will also be an option to remove empty folders in the route section.

SCAN OPTIONS	DESCRIPTION
Scan	Scan all files and folders in scan path. Not recursive.
Scan files only	Scan files only in scan path. Ignore folders. Not recursive.
Scan folders only	Scan folders only in scan path. Ignore files. Not recursive.
Scan recursively	Scan path recursively for files and ignore folders.
Scan/Search Path	The path where to scan for files
Limit to pattern	Regular expression that needs to be matched on file or foldername.
Restrict to same value in dbfield	Only include files with the same value in this database field. Type in name of single database field. EnRoute will check the value of the first file/folder it finds and then only allow new files/folders with same value.
Minimum number of files	The event will only run if the minimum number of files/folders is met.
Maximum number of files	The event will only include this number of files/folders.

4.2.2 Search database using Keywords

Searching the database can be done in two ways: Keyword query and Template-based query. They both use the same selected scan type : **Search database using keywords**

SCAN OPTIONS	DESCRIPTION
Search database using keywords	Query files in the Xinet database and search for specific data assigned to the files.
Scan/Search Path	The base path where to restrict the database query to
Include subfolders in search	Select <i>One level only</i> to search in search path only or <i>Include subfolders</i> to search recursively beneath scan path
Keyword query	Semi-SQL call to use for the query or filename for custom query based on file. The query string can include tokens for keywords, etc. Example: #customer_no#=1234 AND #status#="delivered" See more details and examples below.
File/Folder matching	File scanning pattern to catch names, part of names or extensions. Use % as wildcard, example: %.jpg
Include files and/or folders	Select whether to search for files, folders or both
Minimum number of files	The event will only run if the minimum number of files/folders is met.
Maximum number of files	The event will only include this number of files/folders.

When using a Keyword query, the query string itself is typed into the **keyword query** field. The syntax used is the keyword comparison part of the sql query. It can include one or more keyword names (embedded within # characters) along with the corresponding test criteria, together with logic operators like AND and OR. Example:

```
#CustomerID#="ABC"
```

EnRoute interprets the string and builds the complete query needed to get all the files that meet the search criteria, incorporating the other configurations like File/Folder matching and Search path.

A **Keyword** has to match the name of an existing Xinet Keyword and it need to be embedded within # (hash) characters

The **Operators** that can be used are: = (equal), < (less), > (more), <> (not like) or != (not like).

A **Value** can be boolean (0 or 1) , numerical, string (quoted) or a date operator #vdate()#, #date()# or #lexec()#.

The date operators (date and vdate) can take an argument that will offset the date by a certain number of days forward or backward. vdate() always use the time of day = 12.00 (noon) like the Xinet database when not specifying times. lexec() use the time of the last execution. The format

used for the search is the standard MySQL date. If another format is needed, specify format as an argument.

Example searches:

Searching for keyword CustomerID equal to "ABC"

```
#CustomerID#="ABC"
```

Searching for keyword Project_status equal to "Done" and Done_date older than a week ago

```
#Project_status#="Done" AND #Done_date#<#vdate(-7)#
```

Use the **Test Event** button at the bottom of the page to see whether a search setup result in any files. Type in the search, save and then run Test Event. A window opens which shows detailed login info of the event. The complete search query is shown and also the resulting file list. When using the **Test Event** button any route method is simulated. It will not be executed.

Example. Simple query with two fields, the result and the files being searched on.

Top Level / TEST / SEARCHTEST-1 / 1 (5 matches found)

File Name	Thumbnail	Metadata
0006831.jpg		xtestdate: January 22, 2016 xtestint: 123 xteststr: A string xtestbool: true
0006835.jpg		xtestdate: March 1, 2016 xtestint: 123 xteststr: A string xtestbool: true
0006836.jpg		xtestdate: March 1, 2016 xtestint: 123 xteststr: A string xtestbool: true
0006836.jpg		xtestdate: January 22, 2016 xtestint: 123 xteststr: A string xtestbool: true
0006839.jpg		xtestdate: January 22, 2016 xtestint: 123 xteststr: A string xtestbool: true

There are two fields being used in the search: xtestdate and xtestbool

Scan for files

Scan method: Search database using keywords

Scan/Search path: /home/FPVOLS/TEST/SEARCHTEST-1/1 Include subfolders

Keyword query: #xtestbool#=1 AND #xtestdate#<#vdate0#

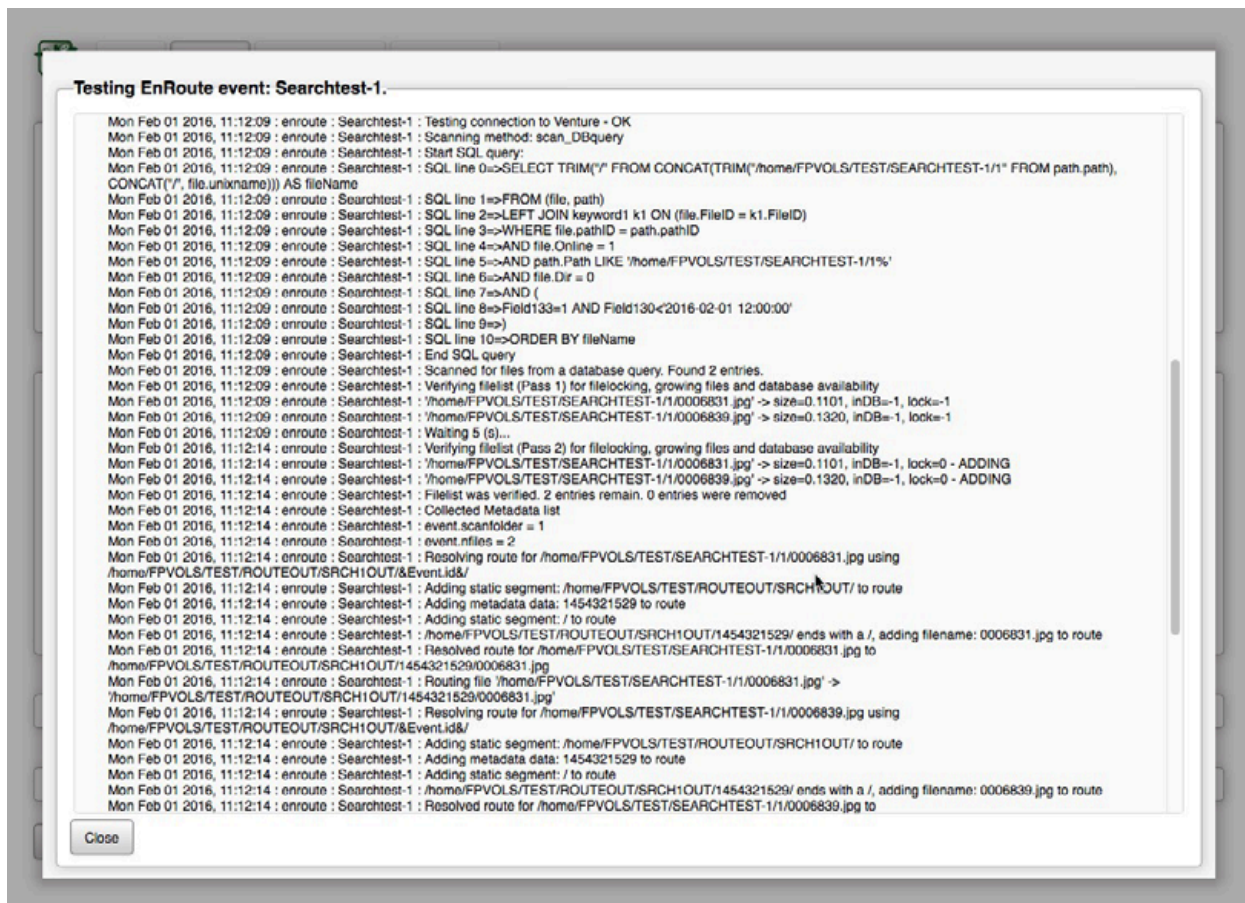
File/Folder matching: File/folder name search pattern Only match files in search

Minimum/Maximum files: Min Max

Run event with no files found: No

Scan wait (s): Custom wait time in seconds

The search is made for assets where xtestbool is set to true (1) and where the datefield xtestdate is before today's date



The Test Event dialogue show the resulting query and searchresult

4.2.3 Search database using Template

Searching the database can be done in two ways: Keyword query and Template-based query. They both use the same selected scan type : **Search database using keywords**

If there is a need to use a more complex search or a search using other parameters than keywords it is possible to do so by using a template that is saved as a file. The query template should be saved as an isolatin 1 file, using UNIX linebreaks. The file can be saved anywhere on the server. The full path to the file is specified the Keyword query setup field. When EnRoute reads a query that starts with “/” it automatically assumes that it is a filebased query.

SCAN OPTIONS	DESCRIPTION
Search database using key-words	Query files in the Xinet database and search for specific data assigned to the files.
Scan/Search Path	The base path where to restrict the database query to
Include subfolders in search	Select <i>One level only</i> to search in search path only or <i>Include subfolders</i> to search recursively beneath scan path
Keyword query = Path to search template	Path to the query template to be used. The path can be followed by optional arguments that can be used in the template. Arguments are separated by semi colon ";" example: /usr/inpress/enroute/setup/queries/myquery;123;158
File/Folder matching	File scanning pattern to catch names, part of names or extensions. Use % as wildcard, example: %.jpg
Subscan method	Select whether to search for files, folders or both
Minimum number of files	The event will only run if the minimum number of files/folders is met.
Maximum number of files	The event will only include this number of files/folders.

The queryfile needs to contain the full query to be passed to the database. This query needs to be designed to return the subpath + filename to the entries found starting at the selected scanpath.

The query needs to be a working MySQL query and it can include some specific tokens:

#SCANPATH#, #ARG0#, #ARG1#

#SCANPATH# translates to the scanpath of the Event.

#ARG0#, ARG1# and so on translate to specific arguments that can be entered in the admin after the actual path to the queryfile.

The arguments are separated by “;” In addition to the already mentioned tokens, #date()#, #vdate()# and #lexec()# can be used as with simple queries.

Example entry in setup:

SCANPATH: /raid/files

DATABASE QUERY: /usr/inpress/enroute/setup/queries/myquery;123;158;159

The contents of the file “myquery” is:

```
SELECT TRIM("#SCANPATH#/" FROM CONCAT(path.Path,"/",file.unixname)) FROM (file,path)
LEFT JOIN keyword1 k1 ON (file.FileID=k1.FileID) WHERE file.pathID=path.pathID
AND file.Online=1
AND file.Dir=0
AND path.Path LIKE '#SCANPATH#/%'
AND Field#ARG0#>'#Date(-28)#'
ORDER BY Field#ARG1#, Field#ARG2#;
```

The query will be (if executed on 2016-01-30):

```
SELECT TRIM("/raid/files" FROM CONCAT(path.Path,"/",file.unixname)) FROM (file,path)
LEFT JOIN keyword1 k1 ON (file.FileID=k1.FileID) WHERE file.pathID=path.pathID
AND file.Online=1
AND file.Dir=0
AND path.Path LIKE '/raid/files'
AND Field123>'2016-01-30'
ORDER BY Field158, Field159;
```

4.2.4 Scan Database content in named folders

Scanning using the **Scan Database content in named folders** method uses the database to find all folders of a certain name that exists under a specific basepath. Then proceed to add all the files/folder from within those folders. It is also possible to add an additional keyword restriction on the search.

SCAN OPTIONS	DESCRIPTION
Scan database content in named folder	Query files in the Xinet database and search for files/folders inside named Parent folder(s). Combine with keyword search if applicable.
Base scan path	Limit the database search to this base path
Parent folder name	The search will match files and folders at the first level within parent folders with this name.
Subscan method	Select whether to find files, folders or both
Additional database query	Semi-SQL call to use for the query or filename for custom query based on file. The query string can include tokens for keywords, etc. Example: #jofolder#=1 AND #status#="closed" See more details below.

Minimum number of files	The event will only run if the minimum number of files/folders is met.
Maximum number of files	The event will only include this number of files/folders.

For example, find all folders within folders named "To Archive" under the Xinet volume Jobs. Only get the folders with the keyword Jobstatus set to "Done".

Search database content in named folder with an additional keyword query

A keyword restriction is identical to the keyword search above.

4.2.5 Scan uploadreports

An uploadreport is a text/html file generated by the WebNative upload cgi when uploading files and using specific upload styles. This functionality is not possible to use when uploading files to Xinet via Portal, pilote or the uploader.

SCAN METHODS	DESCRIPTION
Scan uploadreports	Scan for uploadreports. Files specified in report are identified
Scan/Search Path	The path where to scan for upload reports
Limit to pattern	Regular expression pattern to identify uploadreport. Defaults to upload-report.html" when using "Scan uploadreports"

Since this functionality is only available when using the old styles it is no longer being developed in EnRoute for versions newer than 2.X. It is still in the release for backwards compatibility. See the old manual for version 2.X for more information.

4.2.6 Scan XML files

It is possible to read a very specifically formatted XML file and to use the information for routing, keywording etc. The general idea is to scan for the XML file and let EnRoute open and read the file and parse it for scanning information as opposed to getting scan information from the filesystem or Xinet database.

The XML format can be created from another system that provides files along with an XML sidecar or just an XML file specifying what assets to work with. One example of a system that creates this type of file is InPress Systems InterAct. Specifically the AssetRequest plugin.

SCAN METHODS	DESCRIPTION
Scan xmlfile	Scan for xmlfiles. Files specified in the file are identified
Scan/Search Path	The path where to scan for XML files
Scan Pattern	Regular expression that needs to be matched on file or foldername.
XML element for Metadata	Name to be used to identify the Metadata section in the XML file. Default is <i>metadata</i>
XML element for File	Name to be used to identify the File section in the XML file. Default is <i>file</i>
XML element for Filepath	Name to be used to identify the File path section in the XML file. Default is <i>filepath</i>
XML element for Fileid	Name to be used to identify the File id section in the XML file. Default is <i>fileid</i>

The XML file can hold information about the files and fileids along with additional metadata that can be used to calculate routing paths and applied to keywords.

The XML format supported by the EnRoute parse is like the following :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<import>
  <METADATA>
    <Dataelement1>DATA1</Dataelement1>
    <Dataelement2>DATA2</Dataelement2>
    <Dataelement3>DATA3</Dataelement3>
  </METADATA>
  <FILE>
    <FILEPATH>Filepath</FILEPATH>
    <FILEID>Fileid</FILEID>
    <Dataelement1>DATA1</Dataelement1>
    <Dataelement2>DATA2</Dataelement2>
    <Dataelement3>DATA3</Dataelement3>
  </FILE>
  <FILE>
    <FILEPATH>Filepath</FILEPATH>
    <FILEID>Fileid</FILEID>
    <Dataelement1>DATA1</Dataelement1>
    <Dataelement2>DATA2</Dataelement2>
    <Dataelement3>DATA3</Dataelement3>
  </FILE>
</import>
```

The XML element names METADATA, FILE, FILEPATH and FILEID are the default names that EnRoute use when reading. Other names can be configured in the scan setup. Only XML elements are supported as of this release. Information cannot be given in attributes.

The root element can have any name.

EnRoute reads the filepath and fileid entries and use these to get the files. A fileid overrides path. The folder where the XML is found is used as the basepath whenever a relative filepath is given by the filepath element.

The file is read and the data parsed. Metadata fields are added to the internal EnRoute data to be used for routing and reporting as meta.NAME where name is the element name of the metadata element. Additional elements within the file element are added to the internal EnRoute dataset as file.NAME.

Example. The following dataset would be generated from the XML file below assuming the example XML file below is scanned in the path : /raid/incoming/xmlfiles

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<import>
  <metadata>
    <exporter>John Doe</exporter>
    <branch>Data export</branch>
  </metadata>
  <file>
    <filepath>IMG_0641.JPG</filepath>
    <type>Image</type>
    <owner>Internal</owner>
  </file>
  <file>
    <filepath>/raid/Jobs/123/Catalog.indd</filepath>
    <type>Layout</type>
    <version>1.01</version>
    <date>2016-02-14</date>
  </file>
</import>
```

When parsing this XML file, 2 files would be found. The first one has no path so it would be assumed to be in the same directory as the XML file. Following data is available for routing and reporting after the XML has been parsed.

```
meta.exporter = "John Doe"
meta.branch = "Data export"
```

```
File 0:
(/raid/incoming/xmlfiles/)IMG_0641.JPG
file0.type = "Image"
file0.owner = "Internal"
```

```
File 1:
/raid/Jobs/123/Catalog.indd
file0.type = "Layout"
file0.version = "1.01"
file0.date = "2016-02-14"
```

See following sections for more information regarding how the metadata, filedata etc is used for routing and reporting.

4.2.7 Fetch pop mail

The fetch pop mail scan will connect to a pop email server, get the first non-read email and download it to the server. The subject, content and attachments are retrieved from the mail and the attached files count as the "scanned" files.

SCAN METHODS	DESCRIPTION
Fetch pop mail	Get emails using a pop account. Attached files are treated as scanned
Email server	The server to connect to. Prefix server address with pop3:// or pop3s:// (for secure connections)
Username	Username for the account to connect to
Password	Password for the account to connect to
Forward original email	Select whether to forward a stripped mail (no attachments) and the address to forward to.
Email filter by subject	Ignore and delete emails that contain this string in the subject. Several strings can be added using semicolon as separator. The string can use wildcard (*) and single character wildcard (.)
Filter attachments	Ignore and delete attachments that contain this string in the filename. Several strings can be added using semicolon as separator. The string can use wildcard (*) and single character wildcard (.)
Ignore inline attachments	Select yes to ignore and delete inline attachments

The email parser collects the email address, from address, replyto address and subject and makes these available to EnRoute for routing and reporting purposes as

```
meta.to
meta.subject
meta.from
meta.replyto
```

4.2.8 Filelist limitations

The minimum and maximum number of files can be used to control how a scan is made. By setting a minimum number of files, the event can be forced to "wait" until the right number of files can be found before being launched

By setting a maximum number of files the event will just include exactly the number of files needed and leave the rest behind for the next scan.

Setting the same value on both minimum and maximum will force the event to wait until the right number of files have been found and then run on exactly that number.

4.2.9 Using database field for notifications

EnRoute can read database fields on the files that have been found. This feature can be used to read an email address out of a database field to be used for notifications (see following section). However, using this feature may also put some restraints on the filelist itself.

When using a database field to pickup email notification address, ONLY the files with the same email will be included in a specific scan.

For example, using a scanforfiles on a hot folder and reading the notification email from the database field "Notify". If there are four files, two with Notify="admin@company.com", one with Notify="sales@company.com" and one with a blank value in the field, it will result in three different filelists.

4.3 Routing Method

After EnRoute has scanned and identified the files, it will apply the selected routing method from the Event setup. While “Routing” implies moving files, it is used here in a wider meaning and the files do not have to be moved.

The supported routing methods are **Move, Convert, Transfer via ftp, Send link to asset download, Run Custom Program** and **No Routing**. The different methods will be covered in detail below. Some of the options are general.

GENERAL ROUTE OPTIONS	DESCRIPTION
Delete original files	If set to yes, EnRoute will delete the original file that has been copied.
Clean out orphaned subfolders	Using a “recursive scan” combined with “delete original files” may cause empty folders. This setting will clean out these folders when the event has completed
Unzip	Select to unzip files that were part of the scan. EnRoute will check the first level of the zip and if the zip expands to a folder, it will move into that folder and get the individual files/subfolders when the option "ignore single top folder" is selected,
Save original zipfile	Route the zips that are unzipped to this path.

4.3.1 Move files

Move files will copy the files to the routing path. If the “delete original” function is selected, it will be a move. Move can move to a static path or to a dynamic path based on different data collected in the scan or from the individual files.

Files can also be renamed while routing. If the routing path ends with “/” it is assumed that the path is to a directory and the original filename is added to the path.

Move files can route to two different output paths with separate configurations.

ROUTING OPTIONS	DESCRIPTION
Move files	Copy/Move file to target (routing path)
Routing Path	Target Path where to put the files. Can be static or dynamic. Ending with “/” assumes original filename should be used. See more information and examples below for creating dynamic paths.
Set Finder color, routed file	Pulldown to select the finder lable color to be set on routed file.
Reset database fields, routed file	If set to yes, all the entries in the database for the file will be deleted after routing. Note that this does not affect the original file if kept.
Set database fields, routed file	Formulas for setting database fields on the routed file. The syntax for setting database fields is Fieldname1=Value1;Fieldname2=Value2;.. where Fieldname is the name of a valid Xinetfield and value is static or variable. See more information about setting keywords in following sections.
Max files in target dir	Maximum number of files in the directory where files are moved to. When reaching the max number a new directory is created adding a -N (-1, -2, ..) to the directory name.
Routing Character Filter	Routing Character filter is used to filter the characters used in the routing path. There are three default filters; Only lett, num, underscore to underscore - This filter only accept a-z, A-Z, 0-9 _/.. All other signs will be replaced with underscore (_). Only lett, num, underscore remove - This filter only accept a-z, A-Z, 0-9 _/.. All other signs will be removed. Dash to slash - Replaces dash (-) to slash (/). It's possible to create own filters, they should be placed in the folder /usr/inpress/enroute/setup/routetrans.
Routing Filter Option	Routing Filter Option is used to decide what to filter (routing character filter), just filename, just path or both.

ROUTING OPTIONS	DESCRIPTION
Overwrite file	If set to yes, EnRoute will overwrite existing files otherwise a decimal will be appended to the filename: -N (-1, -2, ..)
Inherit permissions	If set to yes, EnRoute will use the permission details of the first encountered parent directory's for any subfolders and files.
Error Path	Path where to put files in case Routing path cannot be resolved or if a copy cannot be made to the target path
Routing path 2 and related	All the above setups for route path are also present for a second route path copy. When using route path 1 & 2 on a file, there will be two copies created from the original.

Routing method “Move files” moves files to a dynamic path including the date. Original files are deleted and the routed files get two database fields set. Filtering on the filename is enabled.

4.3.2 Convert files

Convert will convert the files to the routing path using a conversion setup. The converted file can be sent to a static path or to a dynamic path based on different data collected in the scan or from the individual files.

Files can also be renamed while routing. If the routing path ends with “/” it is assumed that the path is to a directory and the original filename is added to the path.

Convert files also has the ability to create a copy of the original file on a second path. This copy is created using the move files route method with identical setups.

All the options except the two top options are the same for Convert as for Move.

ROUTING OPTIONS	DESCRIPTION
Convert	Convert file to target (routing path)
Conversion option	Setup to convert to. A setup includes fileformat, resolution, color, and more. New conversion setups can be created (see below).
Handling of non-images	How to handle files that can not be converted; Ignore, Copy to routing path, Copy to error path

Converting images to the setup named Thumbnail. The converted images goes to the path /home/webimages followed by a filename given by the database field webname). Non-images and any file errors (for example where the database field has no value) are moved to the path /home/webimages/ERRORS/.

When the route method Convert is selected, the button "Conversions" show up. This button has the same function as the Conversion button at the bottom of the events page, ie it opens a window that displays all conversion setups saved and allows for editing and adding new setups.

A conversion setup includes the following choices:

CONVERSION SETTING	DESCRIPTION
Name	Name to choose from. Can only use a-z, A-Z and numbers
File format	Format of file
ICC Profile	Profile to use for conversion. Using Profiles to choose the color space overrides any color space settings. It also requires the images to be converted, to have a source profile or that there is a default source profiles set by the Xinet setup for the image type. Add new profiles by copying to: /usr/inpress/enroute/setup/iccprofiles
Color space	Color space to use if not using profile.
Resolution (dpi)	Resolution in dpi
Scale method	Inches, Cm, Mm and Pixel will scale to the appropriate finite size as set by Width and Height. % will scale proportionally to the original size using only Width.
Width	Width in the selected Scaling method. Use only numbers and decimal point.
Height	Height in the selected Scaling method. Use only numbers and decimal point.
Padding	When scaling to a set width and height, using Padding means that the image will always be the set size, using padding to fill up empty space.

4.3.3 Transfer files via ftp/sftp

The transfer method will send the files to a remote ftp or sftp server. Either as a zipped archive or as separate unzipped files.

ROUTING OPTIONS	DESCRIPTION
Ftp/Sftp connection	Transfer files to an ftp or sftp server
Remote server to connect to	Remote server address of the type: ftp://host or sftp://host The address need to start with ftp:// or sftp:// and cannot end with / Example: ftp://ftp.inpress.se
Remote folder	A remote folder can be created automatically for placement of the files that are transferred. It can be one or more folders deep and can contain dynamic data. Leave this empty if the transfer should be sent to the top level of the host. Example: <code>date(%Y-%m)&/&event.id</code> would translate to something like 2016-02/1454320817
Username	ftp/sftp username
Password	ftp/sftp password
File handling method	Send each file separately or make a zipfile. When sending separately the full structure of a folder will be recreated on the target host
Name of zip-archive	The name of the zip when using the file handling method zip The name may include dynamic content. The default is the event id + the string "-archive", example: 1454320817-archive.zip
Email subject	This string will generate a token to be used inside a notification email. The notification template to be used have to include this tag: event.subject
Notification options	The routing method for sending sftp/ftp creates a few new notification tokens that can be used in the normal notifications 1 and 2. There is a sample_html_ftpreceipt to illustrate the notification that can be done.

Transfer files also has the ability to create a copy of the original file on a second path. This copy is created using the move files route method with identical setups.

Example: Transferring separate files and sending an email. Remote folder is used with some dynamic information.

The screenshot displays a web-based configuration interface for file routing. It is divided into two main sections: 'Routing Method' and 'Reporting & Notification'.

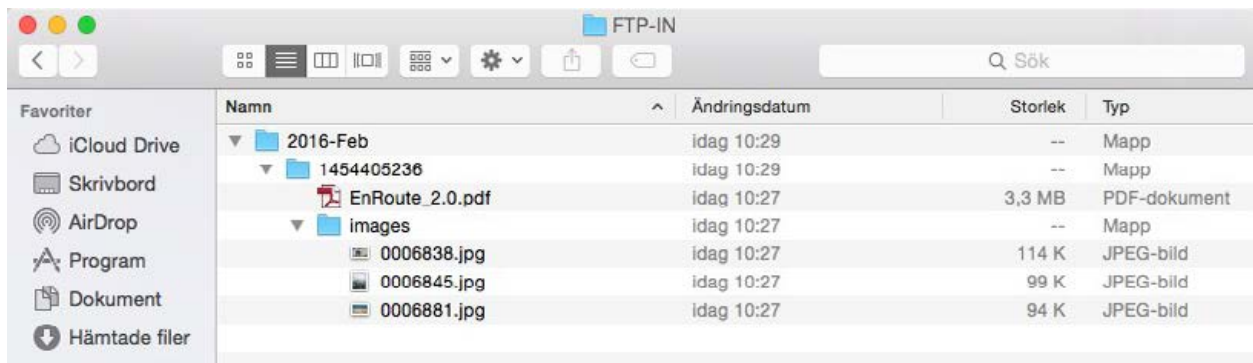
Routing Method Section:

- Routing Method:** A dropdown menu set to 'Transfer files via ftp/sftp'.
- Delete original files:** A dropdown menu set to 'No'.
- Unzip:** A dropdown menu set to 'No'.
- Save original zipfile:** A text input field containing '/path/where/to/store/the/original/zip'.
- Ftp/Sftp connection:** A sub-section containing:
 - Remote server to connect to:** A text input field with 'sftp://192.168.99.111'.
 - Remote folder:** A text input field with 'Users/jorgen/Downloads/FTP-IN/&date(%Y-%m)&/&event.id'.
 - Username and Password:** Two text input fields, the first with 'jorgen' and the second with 'xxxxx'.
 - File handling method:** A dropdown menu set to 'Send each file separately'.
 - Name of zip-archive:** A text input field with 'Archivename without trailing .zip'.
 - Email subject:** A text input field with 'I have sent you files'.
- Second route path / Non-converted route path:** An empty text input field.

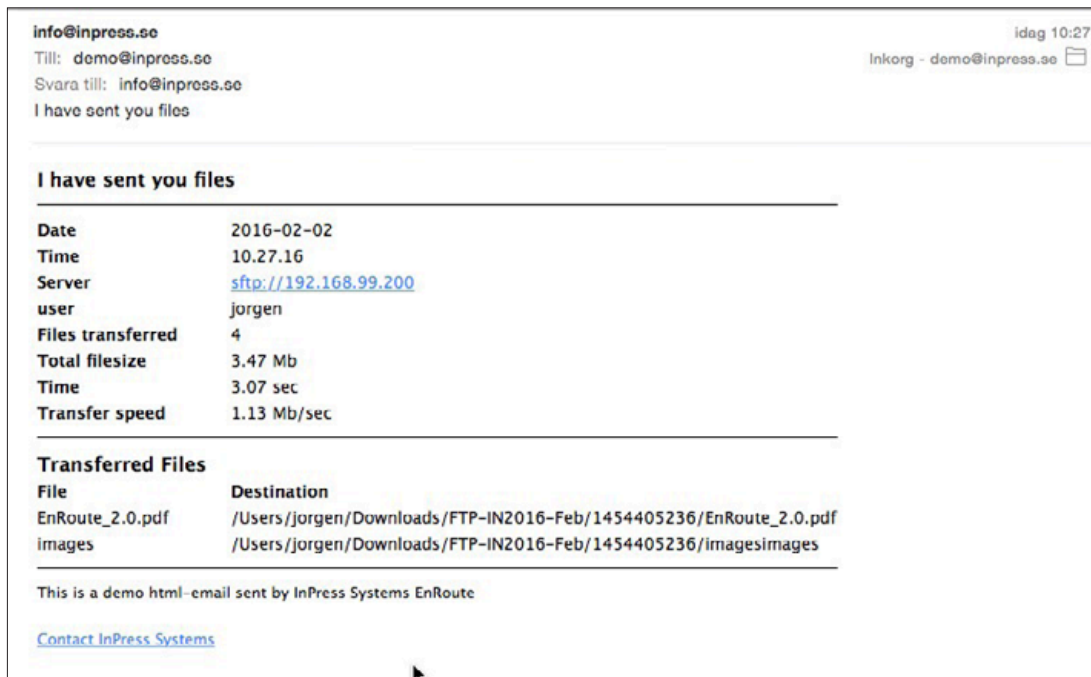
Reporting & Notification Section:

- Notification, email/template:** A text input field with 'demo@inpress.se' and a dropdown menu set to 'sample_html_ftpreceipt'.
- Notification 2, email/template:** A text input field and a dropdown menu set to 'Select template'.
- Mail sender for notifications:** A text input field with 'Defaults to base config: info@inpress.se'.
- Reporting, path/template:** A text input field with '/path/where/to/write/report' and a dropdown menu set to 'Select template'.

The transfer setup sends to an sftp server, using a dynamic remote folder name. Files are sent without zipping.



The resulting folder structure on the ftp server end



The email notification that was sent out using the sample email for transfer

4.3.4 Send link to asset download

Send link to asset download will create a zipped archive and send a link via email to download the file using the Portal interactor cgi. This requires that the Portal server being used has to have the InPress Portal Integration Kit (IPIK) installed.

The method setup is concerned with the zip generation while the notification setup is done under the notifications and reporting section.

Assetlink also has the ability to create a copy of the original file on a second path. This copy is created using the move files route method with identical setups.

ROUTING OPTIONS	DESCRIPTION
Send link to asset download	Create a zip archive and send email with link for download
Name of zip-archive	The name of the zip when using the file handling method zip The name may include dynamic content. The default is the event id + the string "-archive", example: 1454320817-archive.zip
Local path	The local path where to place the temporary zip. It will be deleted automatically when the link expires
Password protect link	Insert a password to protect the link. The word <i>auto</i> will imply automatic generation of the password. Leaving it empty will generate a non-protected link.

ROUTING OPTIONS	DESCRIPTION
Expiration type and length	Expiration in hours. The default is 24 hours. Expire but only use once means that it will expire when used or after the expiration time whatever happens first.
Message	A message that will be displayed when accessing the link and also be in the email that is sent. The email token to be used is: assetlink.message
Email subject	This string will generate a token to be used inside a notification email. The notification template to be used have to include this tag: event.subject

A very important part of the Assetlink method is that an email needs to be sent that includes the link to do the download of the zip. The setup of the email is done using the Notifications and Reports section.

NOTIFICATION OPTIONS	DESCRIPTION
Specific setups for Assetlink	
Notification email	The email to send to
Notification template	The template to use for the notification. The template has to include the right tokens for sending the assetlink. EnRoute includes a sample template: sample_html_assetlink
Mail sender	The replyto address for email. Will default to Base options
Main Server address	Xinet server address. When using a Portal server this needs to be the address for the Xinet server as seen from Portal
Portal Server address	Portal address to be used for the link in the email. The Portal server needs to have IPIK installed.

Example: Setup for sending assetlink.

The screenshot displays the EnRoute configuration interface, divided into two main sections: "Routing Method" and "Reporting & Notification".

Routing Method Section:

- Routing Method:** Set to "Send link to asset download".
- Delete original files:** Set to "No".
- Unzip:** Set to "No".
- Save original zipfile:** Path: /path/where/to/store/the/original/zip
- Assetlink Section:**
 - Name of zip-archive:** Enroute-archive-&event.id
 - Local path:** /home/FPVOLS/TEST/AssetLink-storage
 - Password protect link:** auto
 - Expiration type and length:** Expire, 48
 - Message:** This is a link to download the files for the project.
 - Email subject:** Subject in email when using suitable template. Default: Download assets via link
 - Email setup:** Configure server addresses, email and select an email template suitable for Assetlink in the notifications section below.
- Second route path / Non-converted route path:** (Empty field)

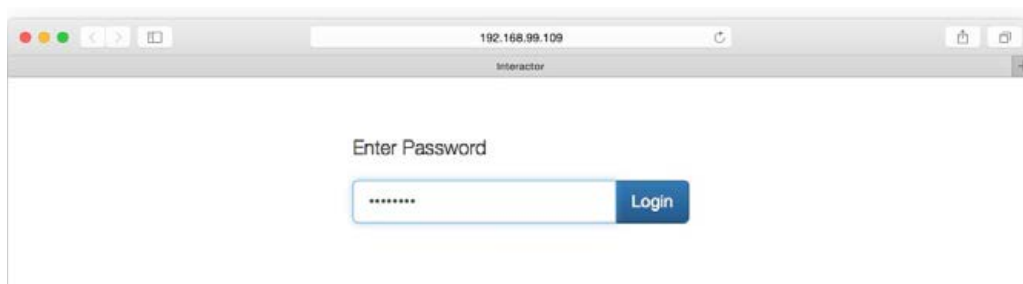
Reporting & Notification Section:

- Notification, email/template:** demo@inpress.se, sample_html_assetlink
- Notification 2, email/template:** (Empty), Select template
- Mail sender for notifications:** Defaults to base config: info@inpress.se
- Reporting, path/template:** /path/where/to/write/report, Select template
- Read Xinet db fields for inclusion:** Fieldname1;Fieldname2;...
- Set database fields, original file:** Fieldname1=Value1;Fieldname2=Value2;...
- Main Server address:** Defaults to base config: http://192.168.99.109:80
- Portal Server address:** http://192.168.99.109
- Portal Sitename:** Sitename

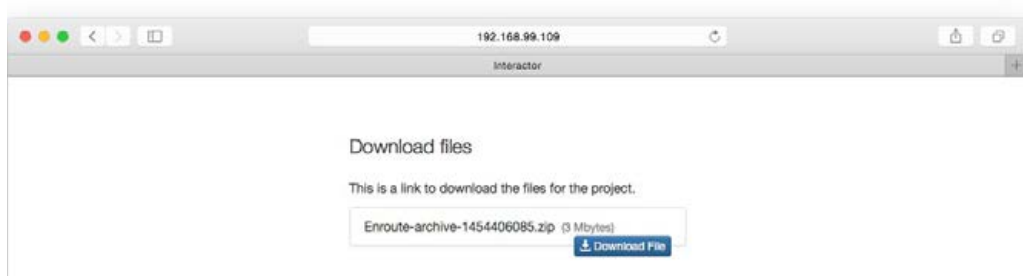
Assetlink configuration along with the corresponding notification. Some setups are from Base configurations.



Email using the assetlink sample template.



Accessing the link



Log in and get the files

4.3.5 Run Custom Program

The method Run Custom Program will run an external program on the files that has been scanned. There are two ways to run the program: **One at a time** or **All at once**.

When running one time per file, the program will execute once per file passing the path to the file as the program's first argument.

When running once for all files, EnRoute will create a temporary file including a list of all the current files. It will execute once while passing the path to the listfile as the program's first argument. After execution, the temp file will be deleted by EnRoute.

ROUTING METHODS	DESCRIPTION
Run Custom Program	Will run an external program.
Custom Program	Write the file name to the external program that is going to be run. Place the external program in the folder /usr/inpress/enroute/setup/extprogs/. Note that the program/script has to be executable.

Custom Argument 1	The first optional argument to the external program
Custom Argument 2	The second optional argument to the external program.
Custom Argument 3	The third optional argument to the external program.
Custom Argument 4	The fourth optional argument to the external program.
Execute type	If there are several files in the file list, to be executed by the custom program, you can choose how to execute them. One at a time - The files will be executed one by one. A path to the file that is going to be executed is sent as argument 1. All at once - All files will be executed at once. A path to a file list is sent to the custom program as argument 1.
Delete original files	If set to yes, EnRoute will delete the original file that has been processed

The custom arguments can be static or dynamic. Dynamic arguments are very similar to the arguments used to create paths. However, when running in the “once for all files” mode it is not possible to use dynamic data from individual files.

Note that the custom script/program will need to handle flags -d (debug), -n (no operation), -f (foreground) when executed. These flags are added on by EnRoute when execution depending on settings and how the event is executed. When running a test, for example the flags -f and -n are added.

4.3.6 No routing

The No routing method is normally used when it is only of interest to use the Event to generate reports, send notifications or to set database fields.

4.3.7 Routing Paths and Custom program arguments

The Routing Path for methods move and convert as well as arguments to running Custom Programs can be static or dynamic. The same goes for some other configurations like the remote folder for ftp and zip archive name in assetlink. Most of the discussion below concerns paths for move/convert but will also be valid for other dynamic arguments.

A dynamic path contains tokens separated by **&** characters. Every token is either a reference to a static path section or to a variable or a function.

Example: `/myRaid/files/Jobs/&db(CustomerName)&/&db(ProjectID)&/`

In the example above we are using the following components:

`/myRaid/files/Jobs/ db(CustomerName) / db(ProjectID) /`

The db() components are dynamic and will get their values querying the Xinet database on the files it's routing. The other components are static.

With the values “ACME” for CustomName and “2010-123456” for ProjectID the path is:

`/myRaid/files/Jobs/ACME/2010-123456/`

The path ends with a “/” meaning that EnRoute will treat this as a path where to put the file using the files original name. It is also possible to create full paths including filename.

Example: `/myRaid/files/SubmittedFiles/&db(ProjectID)&/&date()&/&db(SKU)&file.fext`

In the example above we are using the following static components:

`/myRaid/files/SubmittedFiles/, /, /`

and the following dynamic components:

`db(ProjectID), date(), db(SKU), file.fext`

In this example the resulting path is dependent on the values of two database fields (ProjectID and SKU) as well as the current date. We also pick up the file extension from the original file (if it exists) and adds to the value of the SKU database field which is used as the name.

In this case the path does NOT end with a “/” meaning that EnRoute will treat this as a path to the final file.

There are a number of dynamic sections that can be used based on what the current scenario is. Possible variables to use are values from database fields (from Xinet), data from an upload form (or list), parts of the current filename and specific functions. The dynamic sections can both be used to create routing paths for move/convert and to create arguments for running custom programs, sftp/ftp and assetlink.

DYNAMIC SECTION	DESCRIPTION
file.ATTRIBUTE	File attributes, like “name”, “subname”, “namenoext”, “ext”
file.name	Filename
file.path	Full filepath
file.dir	Path to folder enclosing file
file.namepath	Relative filepath from scan directory to current location
file.namenoext	Filename without any extension
file.ext	File’s extension if any
file.fext	File’s extension if any, including the “.”
file.lastfolder	Name of folder enclosing the file
file.lastfolder_N	Name of Nth folder enclosing the file. N is a number: 1,2,3... 1 means the last folder (ie the same as using file.lastfolder), 2 the second last and so on.
file.fromzip	Name of zipfile where the File was unzipped from. Without “.zip”
file.idx	Index of file inside filelist
file.idx_n	Index of file inside filelist, using n significant numbers. Example: index 5, idx_3 would result in “005”
file.FIELDNAME	Data from file associated uploadform field named FIELDNAME (see uploadform) or file associated data in XML file (when using XML scan)
meta.FIELDNAME	Data from general uploadform field named FIELDNAME or metadata in XML file
event.ATTRIBUTE	Event attributes, like “ID”, “nfiles”, “scanfolder”
event.ID	The event id as specified in setup. Defaults to same value as event.mainID
event.mainID	The internal event id generated by EnRoute. Unix time stamp, 10 digits
event.scanfolder	Name of scanning folder
event.nfiles	Number of current files
date(FORMAT)	Current time expressed in specified FORMAT. EnRoute uses the standard date formatters in UNIX (see appendix for details). If not specified, EnRoute will use. %Y-%m-%d which translates to the YYYY-MM-DD format.
db(FIELD)	Retrieve value for database field FIELD. Error if empty. If field is fid the fileid will be returned.
db2(FIELD;Default)	Retrieve value for database field FIELD. If empty use Default value. If empty and no default, use empty string without error. If field is fid the fileid will be returned.
dbdate(FIELD;FORMAT)	Date stamp from a keyword or the assets Creation, Modification or Access time formatted to specified FORMAT. EnRoute uses the standard date formatters in UNIX (see appendix for details). If not specified, EnRoute will use. %Y-%m-%d which translates to the YYYY-MM-DD format. Specify the Keyword field name or use <i>CreateDate</i> , <i>ModifyDate</i> or <i>AccessDate</i> .
amp	Insert the & character

DYNAMIC SECTION	DESCRIPTION
abr(VALUE;METHOD;ARG)	<p>Abbreviate a value using certain method</p> <p>VALUE=data to abbreviate, for example file.name METHOD=lvl,num,AB,Ab,ab,to,fr ARG=number or digits or characters or token</p> <p>lvl - rounds down to number of digits in ARG: abr(12999;lvl;2) Results in 12000 num - rounds to number of digits in ARG: abr(12999;num;2) Results in 13000 ab - uses number of chars in ARG, lowercase: abr(Wxyz;ab;2) Results in wx Ab - uses number of chars in ARG, no casechange: abr(Wxyz;Ab;3) Results in Wxy AB - uses number of chars in ARG, uppercase: abr(Wxyz;AB;2) Results in WX to - uses string up to ARG: abr(ABC-123;to;-) Results in ABC fr - uses string from ARG: abr(ABC-123;fr;-) Results in 123</p> <p>abr() can be used on most routing tokens including db() and db2()</p>
getsect(VALUE;IDX;SEP) gs(VALUE;IDX;SEP)	<p>Get section of string separated by a separator.</p> <p>VALUE=data to get section from, for example file.name IDX=Letter "L" or number 1 or higher. L will get the last section SEP=the character used as a separator, default is _ (underscore)</p>
lookup(VALUE;FILE;COL) lu(VALUE;FILE;COL)	<p>Lookup a value from a tabdelimited file</p> <p>VALUE=data to use when looking up value, for example file.name FILE=file to use for the lookup. Needs to be in /usr/inpress/enroute/setup/lookup COL=column to read</p> <p>Example: file extmap contains extensions and corresponding names like</p> <pre> jpg webimg psd originals ... </pre> <p>Using lookup(file.ext;extmap;2) will return "webimg" if the extension is jpg</p> <p>lookup() can be used on most routing tokens including db() and db2()</p>
replace(VALUE;FIND;REP) rp(VALUE;FIND;REP)	<p>Replace a substring within a larger string</p> <p>VALUE=string to search and replace in, for example file.dir FIND=section to change (can only be static) REP=replacement text, ie section to change (can only be static)</p> <p>Example: file.dir is /raid1/hotfolders/incoming_jobs Using replace(file.dir;raid1/hotfolders;raid2) will return "/raid2/incoming_jobs"</p> <p>replace() can be used on most routing tokens including db() and db2()</p>
dbd(BASE;F1;V1;F2;V2) dbf(BASE;F1;V1;F2;V2)	<p>Search for a file- or directory path in the Xinet database. dbd restricts to dirpath while dbf restrict to a filepath. dbd() and dbf() returns the shortest path if multiple paths are found.</p> <p>BASE=optional basepath where search should take place. F1=Name of first field to match V1=First value to match F2=Name of second field to match (optional) V2=Second value to match (optional)</p> <p>Search: F1="V1" AND F2="V2" while path LIKE BASE%</p> <p>Note that values can be dynamic. Example: Find project folder that match incoming file's value for project. Add "/uploads/" to path Route: dbd(/raid/projects;PROJID;db(PROJID);PROJFOLDER;1)</p>
upper(VALUE) up(VALUE)	Transform value to uppercase
lower(VALUE) lo(VALUE)	Transform value to lowercase

DYNAMIC SECTION	DESCRIPTION
if(TOK;STR1;STR2)	<p>Test if TOK exist and has a value that is NOT 0.</p> <p>TOK=token to test. STR1=String to use if test is true. STR2=String to use if test is false.</p> <p>example: if(file.ext;upper(file.ext);NOEXT) return the extension as uppercase if it exists, else the string "NOEXT"</p>
if(TOK==VAL;STR1;STR2) if(TOK!=VAL;STR1;STR2)	<p>Test if TOK is equal/not equal to VAL</p> <p>TOK=token to test STR1=String to use if test is true. STR2=String to use if test is false.</p> <p>example: if(file.ext==jpg;IMAGE;OTHER) will return the string IMAGE if extension is jpg, else the string "NOEXT"</p>
customfunc(ARG1;ARG2..)	<p>Run a custom function that will return a path segment. The custom function needs to be an executable file inside /usr/inpress/enroute/setup/extprogs. It can be passed arguments (up to 10) that will be used when the custom function is called. Arguments can be any tokens mentioned in this table. The program/script should return the path segment to be used on stdout. A non-zero exit status indicate an error.</p> <p>Example: /basepath/&myfunc(file.path;date())&/ myfunc will be called and file.path and current date passed as parameters to the script/program.</p> <p>A sample of a custom routing function script is included with the distribution.</p>

Examples using different Route path setups

Example 1:

The goal is to route files that are being uploaded using the Xinet uploader and we want to route based on supplied database information.

Xinet uploader sets two database fields: ProjectID and CustomerName

RoutePath is set to: /myRaid/files/Jobs/&db(CustomerName)&/&db(ProjectID)&/

On an upload using CustomerName: "inpress" and ProjectID: "Enroutemanual" the path would translate to:

Resolved path is: /myRaid/files/Jobs/inpress/Enroutemanual/

The routepath ends with a "/". Enroute will assume this is the directory where to route the file and add the original filename at the end. Routing of a file named "Screenshot" would then place the file in the full path

/myRaid/files/Jobs/inpress/Enroutemanual/Screenshot

Example 2:

Files are uploaded using the WebNative html upload with the House_inpress form. The form has an input field named "CustomerID" and associated to the selected file there is a field named "Status" (for more information about the House_inpress upload style, see a following section).

We want to route the files to an appropriate customer folder, create a subfolder named after today's date and make a subfolder inside that folder for the files status as selected by the person submitting the file.

RoutePath: /myRaid/files/Incoming/&meta.CustomerID&/&date(%Y-%m-%d)&/&file.Status&/

On an upload using CustomerID: “inpress” and Status: “Printready” on Sept 20, 2007 the path translates to

```
Resolved path is: /myRaid/files/Incoming/inpress/2007-09-20/printready/
```

The routepath ends with a “/”. Enroute will assume this is the directory where to route the file to and append the name to the end. Routing of a file named “Enroute manual” would result in the full path looking like this:

```
/myRaid/files/Incoming/inpress/2007-09-20/printready/Enroute manual
```

Example 3:

A folder with files is scanned at a specific time everyday. There are database fields set on the files. We want to use two keywords to file the files into nested subfolders and also rename the file adding _OLD to the filename.

```
Routing Path: /myRaid/files/0ld/&db(Keyword1)&/&db(Keyword2)&/&file.noext&_OLD.&file.fext
```

Processing the file foo.jpg with Keyword1 = “fruit” and Keyword2 = “orange”

```
Translates to: /myRaid/files/0ld/fruit/orange/foo_OLD.jpg
```

The routepath does NOT end with a /. Enroute will assume this is the full path where to route the file including the filename. We are using the tokens file.namenoext and file.ext to generate a new filename while making sure that we preserve any extension.

4.3.8 Updating Database Fields

Depending on the routing method, both original and routed files can have database values set by EnRoute. There is one setup for original files (in the repoerting section) and separate setups on first and second route.

The syntax for setting a database fields is

```
Fieldname1[Operator1]Value1;Fieldname2[Operator2]Value2;..
```

where Fieldname is the name of a valid datafield in Xinet, Operator is one of the valid operators (listed below) and the value part can be static information, dynamic information or a valid function.

Example:

```
Delivered=1;Usedate=date();
```

Following operators can be used:

OPERATOR	DESCRIPTION
Field=VALUE	Sets the Field to VALUE. Example: “Submitdate=date()”. The date function inserts the current date.
Field=+	Increments the field by 1. The field can be integer or string. Example: “Usage=+”.
Field=-	Decrements the field by 1.
Field=+VALUE	Increments the field by numerical VALUE.
Field=-VALUE	Decrements the field by numerical VALUE.
Field=!VALUE	Sets to VALUE only if no the database field is empty. Example: “Converted_on=!date()”.
Field=>VALUE	Inserts string VALUE before any existing value and reapplies the string to the field. Example: “Execute_dates=>date()&,”. A string like “2010-10-25,2009-10-15,2010-06-12” will be maintained, with new additions at the START of the string.

OPERATOR	DESCRIPTION
Field=<VALUE	Inserts string VALUE after any existing value and reapplies string to the field. Example: "Events=<event.ID& ". A string like "0001 0002 0003" will be maintained, with new additions at the END of the string.
Field=[VALUE	Adds string VALUE to a list of unique values. Example: "Events=[event.name". A string like "abc def ghi" will be maintained. Field={VALUE Adds string VALUE to a list of unique values, maintains count of each occurrence. Example: "Events={event.name". A string like "abc(2) def(5) ghi" will be maintained.

The dynamic information to be used when setting a database field is similar but not identical to the information accessible when creating routepaths. It is the same as the information available when creating notifications and reports (see following sections) with the difference that the token used for files is simply "file." while reports and notifications have to take into consideration that there is a list or array of multiple files which is addressed using file.N where N is a number. The @ character is used in notification/report templates to generate loops for multiple files.

VARIABLE VALUE	DESCRIPTION
event.ATTRIBUTE	Event attributes, like "ID", "nfiles", "scanfolder"
event.ID	The event id as specified in setup
event.mainID	The internal EnRoute event id. Unix timestamp (10 digits)
event.scanfolder	Name of scanning folder
event.nfiles	Number of current files
event.procsizemb	Total filesize in Mb for all processed files
event.name	The event name
event.date	Current date of execution, YYYY-MM-DD format
event.time	Current time of execution, in HH:MM format
event.edate	Current date of execution, format as selected in base config
event.etime	Current time of execution, format as selected in base config
event.edatetime	Current date and time of execution, format as selected in base config
event.ldate	Current date of last execution, format as selected in base config
event.ltime	Current time of last execution, format as selected in base config
event.ldatetime	Current date and time of last execution, format as selected in base config
meta.FIELDNAME	Data from general uploadform field named FIELDNAME or general data in XML file (when using XML scan)
file.ATTRIBUTE	File attributes, like "name", "subname", "namenoext", "ext"
file.idx (file)	Index in filelist
file.fid	Xinet file id
file.name	Filename
file.namenoext	Filename without extension
file.ext	File extension (without .) in lowercase. Empty for folders
file.type	File / Folder
file.sizemb	File size in Mb
file.path	Original Filepath
file.dir	Original path to parent folder
file.lastfolder	Folder enclosing original file
file.name_u	Filename, URL encoded
file.path_u	Original Filepath, URL encoded

VARIABLE VALUE	DESCRIPTION
file.dir_u	Original path to parent folder, URL encoded
file.path_x	Original Filepath, Xinet encoded
file.path_wnurl	Original Filepath, WebNative URL. Server address from Event/Base config
file.path_portalurl	Original Filepath, Portal URL. Server address from Event/Base config
file.path_xineturl	Original Filepath, Portal or Webnative URL depending on Main/Portal server addresses from Event/Base config
file.tattribution	First Target attribute, ie file.tname, file.text, file.tlastfolder, file.tpath_u etc NOTE: for transfer via ftp/sftp target = the remote server info.
file.tattribution2	Second Target attribute, ie file.tname2, file.text2, file.tlastfolder2, file.tpath2_u...
file.comment	Files comment if any
file.FIELDNAME	Data from file associated uploadform field named FIELDNAME (see uploadform) or file associated data in XML file (when using XML scan)
file.hasmeta	Exists and is set to 1 if there is any file metadata
file.vent.FIELDNAME	Data from keyword field associated to original file. The keywords to collect need to be specifically listed in the reports section of the event setup. This representation is used to get a value from a specific field.
file.ventnameN file.ventcontN	Data from keyword field associated to original file. The keywords to collect need to be specifically listed in the reports section of the event setup. This representation is used to loop all the fields using .ventname@ and .ventcont@
file.hasvent	Exists and is set to 1 if there is any file keyword data collected
amp	Insert the "&" character
hash	Insert the "#" character
date(OFFSET;FORMAT)	Date function using current timestamp. FORMAT is the date formatter to be used. EnRoute uses the standard date formatters in UNIX (see appendix for details). If no format is specified, EnRoute will default to %Y-%m-%d which translates to dates like "2007-09-31" OFFSET is an optional number (positive or negative). If supplied, the date will be offsetted with as many hours.
update(OFFSET)	Date function. Supplies unix timestamp as number. OFFSET is an optional number (positive or negative). If supplied, the date will be offsetted with as many hours.
abr(VALUE;METHOD;ARG)	Abbreviate a value using certain method. For more information see earlier description in section 3.4.5, Routing Paths and Custom program arguments
nl(n)	insert n newlines. 1 is default
pp(PATH)	Transforms a path to a "pretty path" by getting a certain part of the path, and replacing the / for another string. The arguments are PATH=path to work on, example file.path TYPE=the type of transformation, see below ARG=argument for the type, see below REPL=Optional string with which to replace / (slash) Supported types: <i>first</i> , arg is number of folder levels to get from start <i>last</i> , arg is number of folder levels to get from end <i>lastn</i> , like <i>last</i> but suppress last level <i>from</i> , arg is folder level to start from , 1,2,3... <i>fromn</i> , like <i>from</i> but suppress last level <i>fold</i> , arg is folder name to start from, start with - to NOT show start folder <i>foldn</i> , as <i>fold</i> but suppress last level Examples: pp(file.path;last;3; -) - Get last three levels including file, replace "/" with " - " pp(file.path;fold;-/jobs/acme/) - Get path after "/jobs/acme/", keep /
counter(n)	Return the enRoute event counter with n digits. 0-padded at start

VARIABLE VALUE	DESCRIPTION
getsect(VALUE;IDX;SEP) gs(VALUE;IDX;SEP)	Get section of string separated by a separator. VALUE=data to get section from, for example file.name IDX=Letter "L" or number 1 or higher. L will get the last section SEP=the character used as a separator, default is _ (underscore)
lookup(VALUE;FILE;COL) lu(VALUE;FILE;COL)	Lookup a value from a tabdelimited file VALUE=data to use when looking up value, for example file.name FILE=file to use for the lookup. Needs to be in /usr/inpress/enroute/setup/lookup COL=column to read Example: file extmap contains extensions and corresponding names like <pre> jpg webimg psd originals ... </pre> Using lookup(file.ext;extmap;2) will return "webimg" if the extension is jpg
replace(VALUE;FIND;REP) rp(VALUE;FIND;REP)	Replace a substring within a larger string VALUE=string to search and replace in, for example file.dir FIND=section to change (can only be static) REP=replacement text, ie section to change (can only be static) Example: file.dir is /raid1/hotfolders/incoming_jobs Using replace(file.dir;raid1/hotfolders;raid2) will return "/raid2/incoming_jobs"
upper(VALUE) up(VALUE)	Transform value to uppercase
lower(VALUE) lo(VALUE)	Transform value to lowercase
if(TOK;STR1;STR2)	Test if TOK exist and has a value that is NOT 0. TOK=token to test. STR1=String to use if test is true. STR2=String to use if test is false. example: if(file.ext;upper(file.ext);NOEXT) return the extension as uppercase if it exists, else the string "NOEXT"

Let's take a look at a few examples using different database setups.

Example 1:

We want to route files that are being uploaded using the Xinet uploader and the uploaded files should get today's date set into the **Filesubmit date** field and the default value "submitted" set into the **Filestatus** field. Set database field is set to:

```
Filesubmit date=date();Filestatus=submitted
```

Example 2:

Files are uploaded using the a WebNative html upload with the House_inpress form. The form has an input field named "CustomerID" and associated to the selected file there is a field named "Status" (for more information about the House_inpress upload style, see upcoming section).

We want to update the **Filestatus** database field with the information selected for the file in the form, update the **ProjectID** database field with the CustomerID and put today's date set into the **Filesubmit date** field. Set database field is set to:

```
Filesubmit date=date();Filestatus=file.Status;ProjectID=meta.CustomerID
```

4.4 Reporting & notification

After performing the routing of the files, EnRoute generates a list of data that can be fed into notification/receipt e-mail and/or in a written text report. Both the notification and the report are based on templates and they share the same technology for making custom formatted e-mails and reports as all other InPress Products.

New templates (both for emails and reports) can be custom made and added to the system and then selected. See more details on the templating technology below.

In addition to the report and notifications, it is also possible to make a query to a secondary MySQL server. The query is based on a query templatefile (just like custom queries when scanning) and can use the same data as the report/notification.

Following options are available for Notification and Reporting:

REPORTING & NOTIFICATION	DESCRIPTION
Notification email	E-mail where to send notification. Can be a comma separated list. Email address can be picked up from upload form using meta.FIELDNAME, example: meta.EMAIL. The first email address can also be read from Xinet database on the file, using db(FIELDNAME). Note that reading email from database affects the filelist.
Notification Template	Select template to use. The default template will be used if no template has been selected but an e-mail has been supplied.
Notification 2 email	E-mail where to send notification. Can be a commaseparated list. Email address can be picked up from upload form using meta.FIELDNAME, example: meta.EMAIL.
Notification 2 Template	Select template to use. The default template will be used if no template has been selected but an e-mail has been supplied.
Mail sender for notifications	Reply address to use for notifications, for this specific event. The mail sender setup in the general configuration will be used in case no event Mailsender has been supplied.
Reporting path	Path where to write report for Event. This path can be static or include dynamic segments.
Reporting template	Select template to use. The default template will be used if no template has been selected but a report path has been supplied.
Read Xinet db fields for inclusion	Write the keywords in Xinet, needed for reports and notifications, separated with semicolon (;). The keyword names and values will be gathered and included in the array for reports and notifications.
Set database fields, original	Formulas for setting database fields, on the original file. The syntax for setting database fields is Fieldname1=Value1;Fieldname2=Value2;... where Fieldname is the name of a valid Xinetfield and value is static or variable. The variables can either be information collected from an upload form or functions that can be run. More information and examples in previous section.
Main server address	Xinet server address. Used in links to get to files and directories. Also used when generating links for assetlink. This address should be the address from the Portal server to Xinet when a Portal server is used. Defaults to the Base configuration.
Portal server address	Portal server address. Used in links to get to files and directories. Also used when generating links for assetlink. Defaults to the Base configuration.
Portal Sitename	Portal site name. Used for links in emails and reports.
Mail address for Error mails	E-mail where to send Error message. In certain circumstances, EnRoute will send an error mail.
Content array debug	Emails and reports gets their data from an array generated by EnRoute. It is possible to write this array in order to view the data available. Insert a path where to write the file. The file will be written to the path and named EVENTNAME_cont_UNIQUENUMBER.txt.

REPORTING & NOTIFICATION	DESCRIPTION
Report to external MySQL	
Database host	The IP-number to the server, where the external MySQL database is located.
Database Name	The name of the external MySQL databas (schema).
Database User name and password	The user name and password that is needed to log in to the external MySQL database.
Data to POST or insert	Single line SQL or POST that is submitted to the external server. If used, it will override any template below.
Data template	The template that is used to create the DB queries. The SQL queries must be divided by semicolon (;) and every new SQL query must start on a new line. There is a sample template included as default.

4.4.1 Email notifications

Emails can be sent to multiple recipients using a comma to separate the emails. Email addresses can be static or dynamic. There are two different types of dynamic emails supported: from upload form or from database field.

In order to use an email address supplied by an upload form, type in “meta. “the name of the field, example **meta.EMAIL**.

In order to use an email address supplied from a Xinet field on a scanned file, specify the field-name within the db() function, example db(Notificationmail).

Note that when using a databasefield to retrieve the email it is possible to end up with multiple email addresses. In order to handle this situation, EnRoute will limit the current scan to the first value of email address it encounters and ignore files with other values.



*Using database fields
as email address may
split the scan*

4.4.2 Email notification template format

EnRoute emails are based on mailtemplates. The mailtemplates are files saved into a subfolder (notiftempl) in the EnRoute setup directory (/usr/inpress/enroute/setup). Each file in this directory will show up as a selectable choice in the pulldowns for *Template for first notification* and *Template for second notification*. The reason that there are two emails and two templates is to make it possible to send different emails to different recipients, for example a receipt to someone uploading a file and a report to the admin.

EnRoute ships with a default emails formatted as text only. This template will be used in case a notification email address has been entered but no template selected (or default selected).

To create a new email template, copy the template that needs customization into the the folder. Edit the emails as needed and select it from the admin interface. Please note that any standard emails shipped with EnRoute should not be edited since they may be overwritten when a software upgrade is made.

The email templates are plain text files and should be edited using a text editor like BBEdit or TextWrangler. Save the files using UNIX linefeeds and encoded as ISO Latin 1. Any empty line in the file will be ignored so if you need a line that show up as empty, use a space on that line.

The first line in the email template is used as the Subject line of the email and will be stripped out and not be part of the email itself. If the email is of HTML type it is of utmost importance that the first line after the subject is an <HTML> tag.

```

Uploadreport
<HTML>
<HEAD>
<STYLE TYPE="text/css">

```

The example above is the top four lines of an html email template. "Uploadreport" is the subject and the first line is an HTML doc tag.

EnRoute parses the email template looking for markers specifying tokens that will be exchanged for the unique data pertaining to the specific event. The marker used in the email templates is "#" (the hash character). The token is normally embedded inside two markers: "#event.date#".

```
Enroute Report - #event.name# - #event.date# / #event.time#
```

The section above is taken from the default notification template. It has three tokens: *event.name*, *event.date* and *event.time*, all embedded within "#" characters.

EnRoute builds an array with many different values that come from the current run of the event. The array is structured into sections like *event* and *meta* with subvalues like *event.name* and *event.date*. To insert a value from a certain variable, use the token within the markers.

In addition to the simple sections there are also numbered sections and subsections that can have different length depending on the data available. Example: *file0*, *file1*, *file2* and so on with corresponding entries for *file0.idx*, *file0.name* and so on.

Retrieving data from the numbered sections has to be done using loops. There are two types of loops: **single-line** and **multi-line**. A single-line loop is created by typing one of the numbered variables at a line using a "@" instead of the number. The entire line will be repeated, string with the number "0" and increasing the counter as long as there is data:

```
#file@.idx# : #file@.name#
```

With the data

```

file0.idx = "1"
file0.name = "Acme industries.eps"
file1.idx = "2"
file1.name = "ACME_hq.psd"
file2.idx = "3"
file2.name = "Sale 2010.xls"

```

The template section will be expanded to:

```

1 : Acme industries.eps
2 : ACME_hq.psd
3 : Sale 2010.xls

```

A multi-line loop can be created using specific loop tags. These tags are only used to create the loop and will not be printed. A loop tag needs to be aligned to the left of the line and the following syntax is used: "#TOKEN@<#" where TOKEN is the numbered variable token name - for example "file" - followed by the section to loop and then closing the section again by the end token "#TOKEN@>#".

Following example from the default template shows the use of a multi-line loop on the meta@ variable:

```

#meta@<#
-----
#meta@.name# = #meta@.val#
#meta@>#

```



A multi-line loop tag needs to start the line to the left

With the data

```
meta0 = "0"
meta0.name = "Company"
meta0.val = "ACME Inc"
meta1 = "1"
meta1.name = "Reference"
meta1.val = "Donald Duck"
meta2 = "2"
meta2.name = "Delivery"
meta2.val = "ASAP"
```

The template section will be expanded to:

```
-----
Company = ACME Inc
-----
Reference = Donald Duck
-----
Delivery = ASAP
```

It is possible to put single line loops within multi-line loops. For example, when retrieving Xinet data from files in an event, these turn up as subindex for the main index:

```
file0 = "Acme industries.eps"
file0.idx = "1"
file0.name = "Acme industries.eps"
file0.ventname0 = "Creation date"
file0.ventcont0 = "2010-06-01"
file0.ventname1 = "Artist - Photographer"
file0.ventcont1 = "DD"
file1 = "ACME_hq.psd"
file1.idx = "2"
file1.name = "ACME_hq.psd"
file1.ventname0 = "Creation date"
file1.ventcont0 = "2009-04-30"
file1.ventname1 = "Artist - Photographer"
file1.ventcont1 = "Mickey Mouse"
```

Using template section:

```
#file@<#
-----
#file@.idx# : #file@.name#
#file@.ventname@# : #file@.ventcont@#
#file@>#
```

Expands to:

```
-----
1 : Acme industries.eps
   Creation date : 2010-06-01
   Artist - Photographer : DD
-----
2 : ACME_hq.psd
   Creation date : 2009-04-30
   Artist - Photographer : Mickey Mouse
```

In addition to the repetition tags, there is a tag for logics. The logics tag is very simple and checks for the existence of a certain variable. EnRoute puts a number of these types of variables



A single-line loop can be made within a multi-line loop

into the array to be able to print certain sections only under certain circumstances. The logics tag is almost identical to the loop tag and also needs to be placed to the left on the line. It will not be printed and also has a start and end version: “#TOKEN?<#” and “#TOKEN?>#”

The meta data section is available when EnRoute has parsed an upload using the uploadform parsing. If it finds metadata, the control token *hasMeta* will be available. Using a logics token with *hasMeta* makes it possible to decide whether to print or not print a section:

```
#hasMeta?<#
Supplied information:
#meta@.name# = #meta@.val#

#hasMeta?>#
```

With the data

```
meta0 = “0”
meta0.name = “Company”
meta0.val = “ACME Inc”
meta1 = “1”
meta1.name = “Reference”
meta1.val = “Donald Duck”
meta2 = “2”
meta2.name = “Delivery”
meta2.val = “ASAP”
hasMeta = “1”
```

Following is generated:

```
Supplied information:
Company = ACME Inc
Reference = Donald Duck
Delivery = ASAP
```

If *hasMeta* does not exist in the array, the section will not generate any output. Note that the value of *hasMeta* is not important. It is the fact that it exists at all that makes the test true.

4.4.3 Email notification data and tokens

As per last section, EnRoute generates a large set of data to be used in email notifications and reports. This list of data is generated automatically and “fed into” the notification and report templates. The content is unique for each execution of every event.

It is possible to output the complete list to a text file to see the exact data available for email notifications and reports when running a certain event. To do this, type in an accessible path in the *Content array Debug* setting on the *Reporting & Notification* setup page.

The most common variables are listed in the table below. Note that some data can be accessed from multiple names. In the table all the numbered variables are noted as tokenN, where N will be replaced by 0,1,2,... for example. In some cases there are subsets of numbered variables (as with fileN). These are noted as tokenK.

The main bulk of the dataset consist of fileinformation. The fileinformation is made up of 5 different file arrays: *file*, *fileP*, *fileE*, *fileF* and *fileV*. All files go into the *file* array. *fileP* is made up of the files that have been processed, *fileE* are files that Errored, *fileF* are files that were filtered (not used as of version 3) and *fileV* are files that were found but not verified.

In the list below, we list all the sub sections for the file only. They do exist for the other file arrays if there is an entry for the file in that array.



Logic tokens are used to decide if to print certain sections or not

TOKEN NAME(S)	DESCRIPTION
event.ATTRIBUTE	Event attributes, like "ID", "nfiles", "scanfolder"
event.ID	The event id as specified in setup
event.mainID	The internal EnRoute event id. Unix timestamp (10 digits)
event.scanfolder	Name of scanning folder
event.nfiles	Number of current files
event.procsizemb	Total filesize in Mb for all processed files
event.name	The event name
event.date	Current date of execution, YYYY-MM-DD format
event.time	Current time of execution, in HH:MM format
event.edate	Current date of execution, format as selected in base config
event.etime	Current time of execution, format as selected in base config
event.edatetime	Current date and time of execution, format as selected in base config
event.ldate	Current date of last execution, format as selected in base config
event.ltime	Current time of last execution, format as selected in base config
event.ldatetime	Current date and time of last execution, format as selected in base config
meta.METANAME	Data from general uploadform field named FIELDNAME or general data in XML file (when using XML scan)
metaN, metaN.name	Data from general uploadform field named FIELDNAME or general data in XML file (when using XML scan). Formatted as an array that can be looped. metaN.name is the fieldname
metaN.val	as above. metaN.val is the value
hasMeta	control variable, set to 1 if there are metafield values
file fileP fileE fileV.attribute	File arrays with data. Three arrays may exist. Use the @ character to loop through all the elements. To get a specific element use the number, example: fileP1.name
file	array of all files, loop using file@
filesA	control variable, exists and set to 1 if there are files in file array
fileP	array of PROCESSED files, loop using fileP@
filesP	control variable, exists and set to 1 if there are files in fileP array
fileE	array of ERROR files, loop using loop using fileE@
filesE	control variable, exists and set to 1 if there are files in fileE array
fileV	array of NOT VERIFIED files, loop using fileE@
filesV	control variable, exists and set to 1 if there are files in fileV array
noFiles	control variable, exists and set to 1 if there are no files in scan
file@.idx (file@)	Index in filelist
file@.fid	Xinet file id
file@.name	Filename
file@.namenoeext	Filename without extension
file@.ext	File extension (without .) in lowercase. Empty for folders
file@.type	File / Folder
file@.sizemb	File size in Mb
file@.path	Original Filepath
file@.dir	Original path to parent folder

TOKEN NAME(S)	DESCRIPTION
file@.lastfolder	Folder enclosing original file
file@.name_u	Filename, URL encoded
file@.path_u	Original Filepath, URL encoded
file@.dir_u	Original path to parent folder, URL encoded
file@.path_x	Original Filepath, Xinet encoded
file@.path_wnurl	Original Filepath, WebNative URL. Server address from Event/Base config
file@.path_portalurl	Original Filepath, Portal URL. Server address from Event/Base config
file@.path_xineturl	Original Filepath, Portal or Webnative URL depending on Main/Portal server addresses from Event/Base config
file@.tattribute	First Target attribute, ie file.tname, file.text, file.tlastfolder, file.tpath_u etc NOTE: for transfer via ftp/sftp target = the remote server info.
file@.tattribute2	Second Target attribute, ie file.tname2, file.text2, file.tlastfolder2, ...
file@.comment	Files comment if any
file@.FIELDNAME	Data from file associated uploadform field named FIELDNAME (see uploadform) or file associated data in XML file (when using XML scan)
file@.hasmeta	Exists and is set to 1 if there is any file metadata
file.vent.FIELDNAME	Data from keyword field associated to original file. The keywords to collect need to be specifically listed in the reports section of the event setup. This representation is used to get a value from a specific field.
file.ventnameN file.ventcontN	Data from keyword field associated to original file. The keywords to collect need to be specifically listed in the reports section of the event setup. This representation is used to loop all the fields using .ventname@ and .ventcont@
file.hasvent	Exists and is set to 1 if there is any file keyword data collected
Functions	Functions that use values from arrays
amp	Insert the "&" character
hash	Insert the "#" character
date(OFFSET;FORMAT)	Date function using current timestamp. FORMAT is the date formatter to be used. EnRoute uses the standard date formatters in UNIX (see appendix for details). If no format is specified, EnRoute will default to %Y-%m-%d which translates to dates like "2007-09-31" OFFSET is an optional number (positive or negative). If supplied, the date will be offsetted with as many hours.
udate(OFFSET)	Date function. Supplies unix timestamp as number. OFFSET is an optional number (positive or negative). If supplied, the date will be offsetted with as many hours.
abr(VALUE;METHOD;ARG)	Abbreviate a value using certain method. For more information see earlier description in section 3.4.5, Routing Paths and Custom program arguments
nl(n)	insert n newlines. 1 is default

TOKEN NAME(S)	DESCRIPTION						
pp(PATH)	<p>Transforms a path to a "pretty path" by getting a certain part of the path, and replacing the / for another string.</p> <p>The arguments are PATH=path to work on, example file.path TYPE=the type of transformation, see below ARG=argument for the type, see below REPL=Optional string with which to replace / (slash)</p> <p>Supported types: <i>first</i>, arg is number of folder levels to get from start <i>last</i>, arg is number of folder levels to get from end <i>lastn</i>, like <i>last</i> but surpress last level <i>from</i>, arg is folder level to start from , 1,2,3... <i>fromn</i>, like <i>from</i> but surpress last level <i>fold</i>, arg is folder name to start from, start with - to NOT show start folder <i>foldn</i>, as <i>fold</i> but surpress last level</p> <p>Examples: pp(file.path;last;3; -) - Get last three levels including file, replace "/" with " - " pp(file.path;fold;-/jobs/acme/) - Get path after "/jobs/acme/", keep /</p>						
counter(n)	Return the enRoute event counter with n digits. 0-padded at start						
getsect(VALUE;IDX;SEP) gs(VALUE;IDX;SEP)	<p>Get section of string separated by a separator.</p> <p>VALUE=data to get section from, for example file.name IDX=Letter "L" or number 1 or higher. L will get the last section SEP=the character used as a separator, default is _ (underscore)</p>						
lookup(VALUE;FILE;COL) lu(VALUE;FILE;COL)	<p>Lookup a value from a tabdelimited file</p> <p>VALUE=data to use when looking up value, for example file.name FILE=file to use for the lookup. Needs to be in /usr/inpress/enroute/setup/lookup COL=column to read</p> <p>Example: file extmap contains extensions and corresponding names like</p> <table> <tr> <td>jpg</td> <td>webimg</td> </tr> <tr> <td>psd</td> <td>originals</td> </tr> <tr> <td>...</td> <td></td> </tr> </table> <p>Using lookup(file.ext;extmap;2) will return "webimg" if the extension is jpg</p>	jpg	webimg	psd	originals	...	
jpg	webimg						
psd	originals						
...							
replace(VALUE;FIND;REP) rp(VALUE;FIND;REP)	<p>Replace a substring within a larger string</p> <p>VALUE=string to search and replace in, for example file.dir FIND=section to change (can only be static) REP=replacement text, ie section to change (can only be static)</p> <p>Example: file.dir is /raid1/hotfolders/incoming_jobs Using replace(file.dir;raid1/hotfolders;raid2) will return "/raid2/incoming_jobs"</p>						
upper(VALUE) up(VALUE)	Transform value to uppercase						
lower(VALUE) lo(VALUE)	Transform value to lowercase						
if(TOK;STR1;STR2)	<p>Test if TOK exist and has a value that is NOT 0.</p> <p>TOK=token to test. STR1=String to use if test is true. STR2=String to use if test is false.</p> <p>example: if(file.ext;upper(file.ext);NOEXT) return the extension as uppercase if it exists, else the string "NOEXT"</p>						

Two of the routemethods, sendbyftp and assetlink, generate specific tokens for emails, only available for these methods. There are sample email templates included in the distribution which illustrate how to use these tokens. The lists of tokens are as follows

TOKEN NAME(S)	DESCRIPTION
For ftp/sftp method	The tokens can only be used in notifications when using the ftp/sftp method
event.subject	Email subject
event.transServer	Server sent to
event.transUser	Username used when sending the files
event.transNumberFiles	Number of files being sent
event.transFSize	Total filesize in Mb for all sent files
event.transTime	Total time used when sending
event.transSpeed	Transfer speed

TOKEN NAME(S)	DESCRIPTION
For ftp/sftp method	The tokens can only be used in notifications when using the ftp/sftp method
assetlink.subject	Email subject
assetlink.message	Email message
assetlink.expiresHours	Number of hours from creation to expiration
assetlink.expiresDate	Date and time for expiration
assetlink.md5	md5 checksum for the zip file
assetlink.password	Password to access the download

The table below describes the different types of tokens that can be used in an email or report template.

TOKEN TYPE	DESCRIPTION
#NAME#	Simple token to be replaced.
##	Replace with # (single #)
#Function()#	Replace with value of Function, see functions above
#NAME@#	Repeatable token. Will repeat for data NAME0, NAME1, ...
#NAME@<# ... #NAME@>#	Repeatable token section. Will repeat entire section between start and end token for data NAME0, NAME1, ... and replace each section with the corresponding values of NAME0, NAME1, ... <i>Tokens need to be aligned to left in file</i>
#NAME?<# ... #NAME?>#	Logic token. <i>Tokens need to be aligned to left in file</i> The section between start and end token will be processed if data NAME exists and has a value.
#!NAME?<# ... #NAME?>#	Logic token. <i>Tokens need to be aligned to left in file</i> The section between start and end token will be processed if data NAME does not exist or exists but has no value.
#NAME==VALUE?<# ... #NAME?>#	Logic token. <i>Tokens need to be aligned to left in file</i> The section between start and end token will be processed if data NAME exists and has a value exactly equal to VALUE
#NAME!=VALUE?<# ... #NAME?>#	Logic token. <i>Tokens need to be aligned to left in file</i> The section between start and end token will be processed if data NAME does not exist or exist and has a value not equal to VALUE

Email Example:

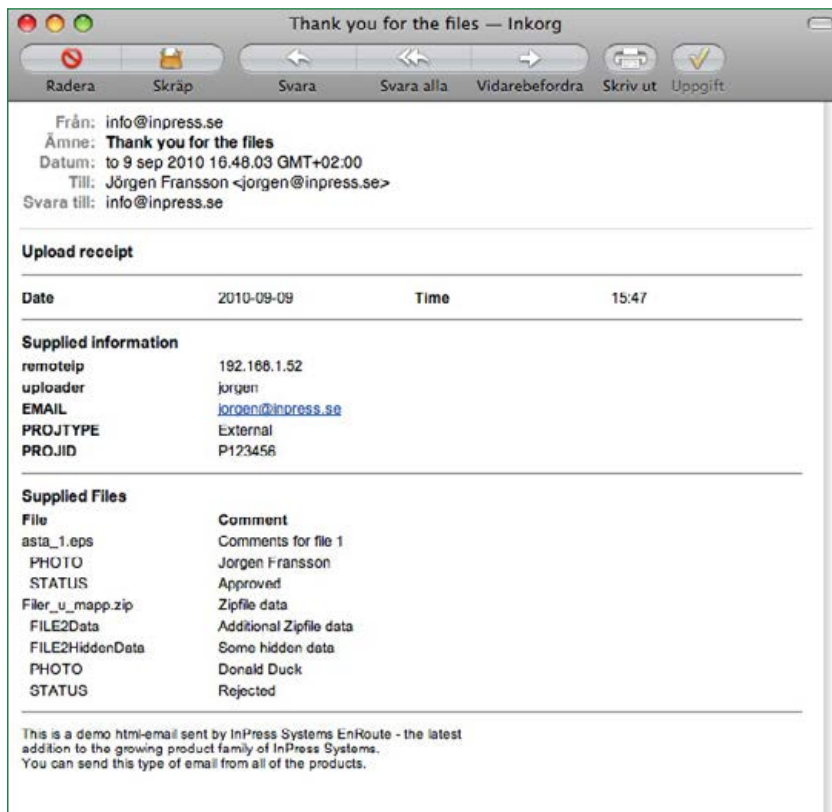
```

sample_emailreceipt
Last Saved: 2010-09-09 13:02:00
File Path: ftp://root@192.168.1.101/usr/enroute/setup/notiftempl/sample_emailreceipt

1 Thank you for the files
2 <HTML>
3 <HEAD>
4 <STYLE TYPE="text/css"> </STYLE>
45 <TITLE>Upload receipt</TITLE>
46 </HEAD>
47
48 <!--
49
50     This is a sample emailtemplate to be used as a receipt for uploads using uploadform
51
52 -->
53
54 <BODY BGCOLOR="white" LINK="black" VLINK="black" ALINK="black">
55
56 <TABLE CELLSPACING=2 CELLPADDING=0 BORDER=0>
57
58 <TR><TD class='headline' colspan=4>Upload receipt</TD></TR>
59
60 <TR><TD colspan=4><HR COLOR="black" SIZE=1 noshade></TD></TR>
61
62 <TR>
63     <TD class='boldtext' width=25%>Date</TD><TD class='text' width=25%>#enroute.date#</TD>
64     <TD class='boldtext' width=25%>Time</TD><TD class='text' width=25%>#enroute.time#</TD>
65 </TR>
66
67 <!-- this section will print only if there are metafields -->
68 #hasMeta?<#
69 <TR><TD colspan=4><HR COLOR="black" SIZE=1 noshade></TD></TR>
70 <TR><TD class='headline' colspan=4>Supplied information</TD></TR>
71
72 <!-- this section will repeat for every extra field on the uploadform, ie the metafields -->
73 <TR><TD class='boldtext' width=25%>#meta@.name#</TD><TD class='text' colspan=3>#meta@.val#</TD></TR>
74 #hasMeta?>#
75
76 <TR><TD colspan=4><HR COLOR="black" SIZE=1 noshade></TD></TR>
77 <TR><TD class='headline' colspan=4>Supplied Files</TD></TR>
78
79 <TR>
80     <TD class='boldtext'>File</TD>
81     <TD class='boldtext' colspan=3>Comment</TD>
82 </TR>
83
84 <!-- this section will repeat for every file, printing the file name and any filecomment -->
85 <!-- if using file metafields in uploadform, these will also be included -->
86 #fileP@<#
87 <TR>
88     <TD class='text'>#fileP@.name#</TD>
89     <TD class='text' colspan=3>#fileP@.comment#</TD>
90 </TR>
91 #fileP@.hasmeta?<#
92 <TR><TD class='text'>&nbsp;&nbsp;&nbsp;&#fileP@.metaname#</TD><TD class='text' colspan=3>#fileP@.metaval#</TD></TR>
93 #fileP@.hasmeta?>#
94 #fileP@>#
95
96 <TR><TD colspan=4><HR COLOR="black" SIZE=1 noshade></TD></TR>
97
98 <TR><TD class='smalltext' colspan=4 width=100%>This is a demo html-email sent by InPress Systems EnRoute - the latest<br>addition to
99 <TR><TD class='smalltext' colspan=4 width=100%>&nbsp;&nbsp;&nbsp;&</TD></TR>
100
101 </TABLE>
102
103 </BODY>
104 </HTML>

```

Template for generating an email receipt used in an event collecting uploads from form



Resulting email receipt from template above.

4.4.4 Reporting

EnRoute can write a report every time an event is run. The report is written to a specific path and appended to any existing file that may be residing at the path (unless *;replace* is added at the end on the path). The report path can be dynamic and include tokens pointing to information in the event's data (see previous section).

Example: Making a daily report file from an event

Output path for reports: `/myraid/Reports/uploads/&date(%Y%m%d)&.report.txt`

Uploading files where this Event is running on Sept 20, 2010, will cause EnRoute to write reports to

`/myraid/Reports/uploads/20100920.report.txt`

If more than one job is run on the setup the same day, the report will be appended to for every time the Event runs.

Should a report be replaced instead of appended, the path should end with *;replace*.

Example: Making a report file used as an include to show latest addition to webnative

Output path for reports: `/var/adm/webnative/mysite/latestfiles.js;replace`

Every time the Event runs the report file will be replaced instead of appened.

4.4.5 Report template format

EnRoute uses templates to create the reports. The templates share the same technology creating the files as when making email notifications. Further, the same content array is used to "feed" the report template.

The report templates are files saved into a subfolder (reporttempl) in the EnRoute setup directory (/usr/inpress/enroute/setup). Each file in this directory will show up as a selectable choice in the pull downs for *Template for reports*.

EnRoute ships with a default report template. This template will be used in case a report path has been entered but no template selected (or default selected).

To create a new report template, copy the template that needs customization into the the folder. Edit the template as needed and select it from the admin interface. Please note that any standard reports shipped with EnRoute should not be edited since they may be overwritten when a software upgrade is made.

The report templates are in plain text and should be edited using a text editor like BBEdit or TextWrangler. Save the files using UNIX linefeeds and encoded as ISO Latin 1. Any empty line in the file will be ignored so if you need a line that shows up as empty, use a space on that line.

When working with reports there are a couple of differences from working with notifications:

- The first line is part of the report. (For notifications, the first line is used as subject in the mail and removed from the body.)
- It is possible to use a header and a tail section in the report template. The header will only be written when a new file is created. The trailer will be removed and re-added everytime a report file is being appended with new data.

Loops, logics and tokens all work the same way as with notifications.

Header section is defined by:

```
<HEAD>
... more lines here
</HEAD>
```

Tail section is defined by:

```
<TAIL>
... more lines here
</TAIL>
```

Note that the tail cannot include any repetitions.

Example. Report for uploads using an uploadform that has two general fields as well as one metadata field per uploaded file.

Report path:

```
/raid/Reports/Uploads/&date()&.xml
```

Report template:

```
<HEAD>
<?xml version="1.0" encoding="ISO-8859-1"?>
<Uploads>
</HEAD>
<upload id="#event.ID#">
  <uploadinfo>
    id="#event.ID#"
    date="#event.date#"
    time="#event.time#"
    #meta@name#="#meta@.val#"
  </uploadinfo>
  #fileP@<#
```

```

<file idx="#fileP@.idx#">
    name="#fileP@.tname#"
    path="#fileP@.tpath#"
    instruction="#fileP@.instruction#"
</file>
#fileP@>#
</upload>
<TAIL>
</Uploads>
</TAIL>

```

Running the event and catching two uploads on the same date (2010-10-01) may end up creating following file (using some sample data and other setups)

Filename: /raid/Reports/Uploads/2010-10-01.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<Uploads>
<upload id="2010-10-01-0001">
<uploadinfo>
    id="2010-10-01-0001"
    date="2010-10-01"
    time="14.32"
    Company="ACME"
    PO="123456"
</uploadinfo>
<file idx="1">
    name="Businesscard.pdf"
    path="/raid/uploaddata/2010-10-01/0001/Businesscard.pdf"
    instruction="Ready to print file"
</file>
<file idx="2">
    name="Letter.pdf"
    path="/raid/uploaddata/2010-10-01/0001/Letter.pdf"
    instruction="Ready to print file. Please check"
</file>
</upload>
<upload id="2010-10-01-0002">
<uploadinfo>
    id="2010-10-01-0002"
    date="2010-10-01"
    time="19.13"
    Company="ACME Service"
    PO="123458"
</uploadinfo>
<file idx="1">
    name="Productsheet.zip"
    path="/raid/uploaddata/2010-10-01/0002/Productsheet.zip"
    instruction="Zip includes all needed components"
</file>
</upload>
</Uploads>

```

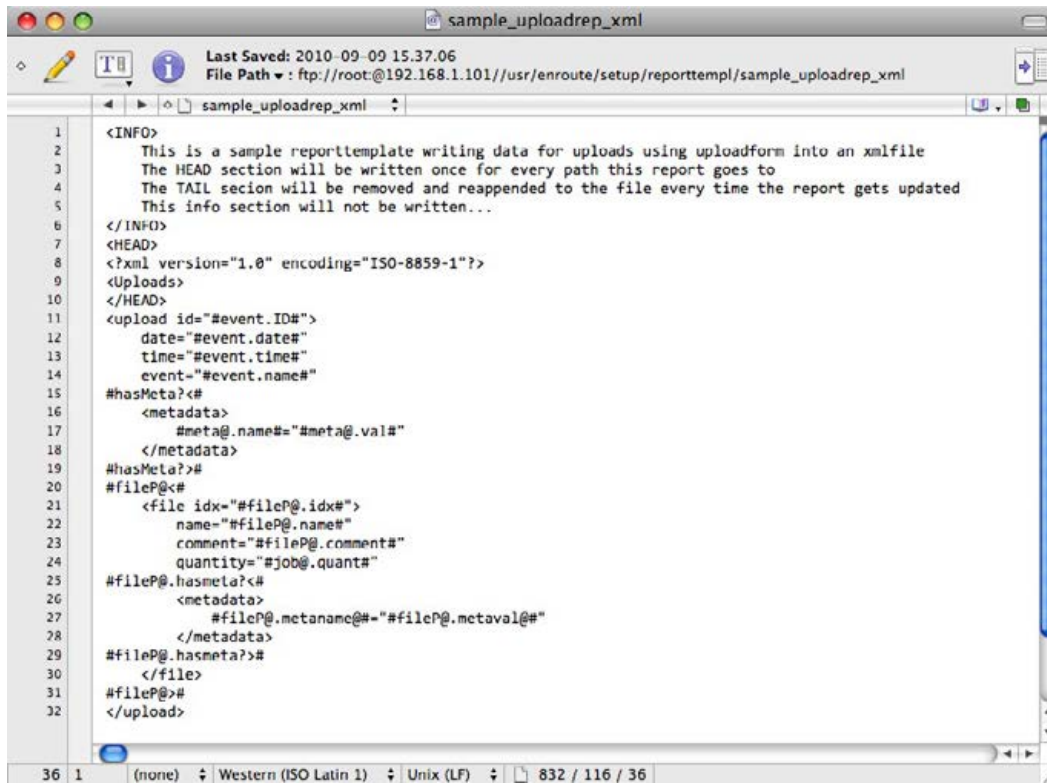
4.4.6 Report data and tokens

Reports share the data and tokens with Email notifications. For an explanation on the different tokens to be used, see “Email notification data and tokens” above.

4.4.7 Report examples

Below are a few examples of reports showing the template and the final result.

First Example: XML report



The screenshot shows a text editor window titled 'sample_uploadrep.xml'. The editor displays an XML template with the following content:

```
1 <INFO>
2   This is a sample reporttemplate writing data for uploads using uploadform into an xmlfile
3   The HEAD section will be written once for every path this report goes to
4   The TAIL section will be removed and reappended to the file every time the report gets updated
5   This info section will not be written...
6 </INFO>
7 <HEAD>
8   <?xml version="1.0" encoding="ISO-8859-1"?>
9   <Uploads>
10  </Uploads>
11  <upload id="#event.ID#">
12    date="#event.date#"
13    time="#event.time#"
14    event="#event.name#"
15    #hasMeta?<#
16      <metadata>
17        #meta@.name#="#meta@.val#"
18      </metadata>
19    #hasMeta?>#
20    #fileP@<#
21      <file idx="#fileP@.idx#">
22        name="#fileP@.name#"
23        comment="#fileP@.comment#"
24        quantity="#job@.quant#"
25      #fileP@.hasmeta?<#
26        <metadata>
27          #fileP@.metaname@="#fileP@.metaval@#"
28        </metadata>
29      #fileP@.hasmeta?>#
30    </file>
31    #fileP@>#
32  </upload>
```

The status bar at the bottom indicates the file is 832 / 116 / 36 bytes, using Western (ISO Latin 1) encoding, Unix (LF) line endings, and is on line 36, column 1.

XML report template. The header is only printed when file is created. The tail (not shown in screenshot) is maintained at the end of the file.

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Uploads>
3   <upload id="1284043902">
4     <date>"2010-09-09">
5     <time>"15:51">
6     <event>"tesuploadreport">
7       <metadata>
8         <remoteip>"192.168.1.52">
9         <uploader>"jorgen">
10        <EMAIL>"jorgen@inpress.se">
11        <PROJTYPE>"External">
12        <PROJID>"P123456">
13      </metadata>
14      <file idx="1">
15        <name>"asta_1.eps">
16        <comment>"Comments for file 1">
17          <metadata>
18            <PHOTO>"Jorgen Fransson">
19            <STATUS>"Approved">
20          </metadata>
21        </file>
22        <file idx="2">
23          <name>"Filer_u_mapp.zip">
24          <comment>"Zipfile data">
25            <metadata>
26              <FILE2Data>"Additional Zipfile data">
27              <FILE2HiddenData>"Some hidden data">
28              <PHOTO>"Donald Duck">
29              <STATUS>"Rejected">
30            </metadata>
31          </file>
32        </upload>
33      </Uploads>
34

```

The final report after the event has been executed once.

Second Example: report used as an included .js file to create a list of last added assets.

```

1 <INFO>
2   This is a sample reporttemplate writing data for a list of files into a js include file
3   The HEAD section will be written once for every path this report goes to
4   It was used to generate a filelist that can be displayed upon login in webnative.
5   The list of files was created using a custom query that found the last N files added to the system in a certain path.
6   A number of datafields was also added to the list
7   This info section will not be written...
8 </INFO>
9 <HEAD>
10  var files = new Array();
11  var i=0;
12  </HEAD>
13  #fileP@<#
14  // Start of filedata
15  files[i] = new Object();
16  files[i].name = "#fileP@.name#";
17  files[i].path = "#fileP@.path_u#";
18  #fileP@.hasvent?<#
19  files[i].keywords = new Array();
20  var j=0;
21  files[i].keywords[j] = new Object(); files[i].keywords[j].name = "#fileP@.ventname@#"; files[i].keywords[j++].value = "#fileP@.
22  #fileP@.hasvent?>#
23  i++;
24  #fileP@>#
25

```

Template to create a js file listing filenames, urls and database values in an array.

```

1  var files = new Array();
2  var i=0;
3  // Start of filedata
4  files[i] = new Object();
5  files[i].name = "Building.jpg";
6  files[i].path = "/FP_vol/Projects/Enroute/Testscan/Other%20files/Building.jpg";
7  files[i].keywords = new Array();
8  var j=0;
9  files[i].keywords[j] = new Object(); files[i].keywords[j].name = "Manufacturer"; files[i].keywords[j++].value = "ACME";
10 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "SKU number"; files[i].keywords[j++].value = "123456789";
11 i++;
12 // Start of filedata
13 files[i] = new Object();
14 files[i].name = "Cars on road.jpg";
15 files[i].path = "/FP_vol/Projects/Enroute/Testscan/Other%20files/Cars%20on%20road.jpg";
16 files[i].keywords = new Array();
17 var j=0;
18 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "Manufacturer"; files[i].keywords[j++].value = "ACME Industries";
19 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "SKU number"; files[i].keywords[j++].value = "123456798";
20 i++;
21 // Start of filedata
22 files[i] = new Object();
23 files[i].name = "House_large.jpg";
24 files[i].path = "/FP_vol/Projects/Enroute/Testscan/Other%20files/House_large.jpg";
25 files[i].keywords = new Array();
26 var j=0;
27 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "Manufacturer"; files[i].keywords[j++].value = "Hobbiton Enterprises";
28 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "SKU number"; files[i].keywords[j++].value = "ABC-123456";
29 i++;
30 // Start of filedata
31 files[i] = new Object();
32 files[i].name = "People.jpg";
33 files[i].path = "/FP_vol/Projects/Enroute/Testscan/People.jpg";
34 files[i].keywords = new Array();
35 var j=0;
36 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "Manufacturer"; files[i].keywords[j++].value = "ACME Industries";
37 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "SKU number"; files[i].keywords[j++].value = "123456798";
38 i++;
39 // Start of filedata
40 files[i] = new Object();
41 files[i].name = "Shoes.jpg";
42 files[i].path = "/FP_vol/Projects/Enroute/Testscan/Shoes.jpg";
43 files[i].keywords = new Array();
44 var j=0;
45 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "Manufacturer"; files[i].keywords[j++].value = "VonDuck Industries";
46 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "SKU number"; files[i].keywords[j++].value = "DUCK-123";
47 i++;
48 // Start of filedata
49 files[i] = new Object();
50 files[i].name = "Exhibition.jpg";
51 files[i].path = "/FP_vol/Projects/Enroute/Testscan/Webimages/Exhibition.jpg";
52 files[i].keywords = new Array();
53 var j=0;
54 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "Manufacturer"; files[i].keywords[j++].value = "VonDuck Industries";
55 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "SKU number"; files[i].keywords[j++].value = "DUCK-456";
56 i++;
57 // Start of filedata
58 files[i] = new Object();
59 files[i].name = "New sales.jpg";
60 files[i].path = "/FP_vol/Projects/Enroute/Testscan/Webimages/New_sales.jpg";
61 files[i].keywords = new Array();
62 var j=0;
63 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "Manufacturer"; files[i].keywords[j++].value = "Hobbiton Enterprises";
64 files[i].keywords[j] = new Object(); files[i].keywords[j].name = "SKU number"; files[i].keywords[j++].value = "ABC-123456";
65 i++;
66 // Start of filedata
67 files[i] = new Object();
68 files[i].name = "Productcell.jpg";
69 files[i].path = "/FP_vol/Projects/Enroute/Testscan/Webimages/Productcell.jpg";
70 files[i].keywords = new Array();
71 var j=0;

```

Final js file listing with filenames, urls and database values in an array.

4.4.8 Connect to external mySQL

EnRoute can be set up to connect to an external mySQL database and run SQL queries using the data from the event.

The query is created from a query template much like the report template. It uses the same data, tokens and technology to render as the notifications and reports. Instead of being sent as email or written to disk, it is used to build the queries to send to the external mySQL server.

Inside the "/usr/inpress/enroute/setup/dbTempl" is a sample template: DB sample, that illustrate how a query file may be setup. The actual file to be used depends on the mySQL database to connect to and its layout.

4.5 Testing and Running events from admin

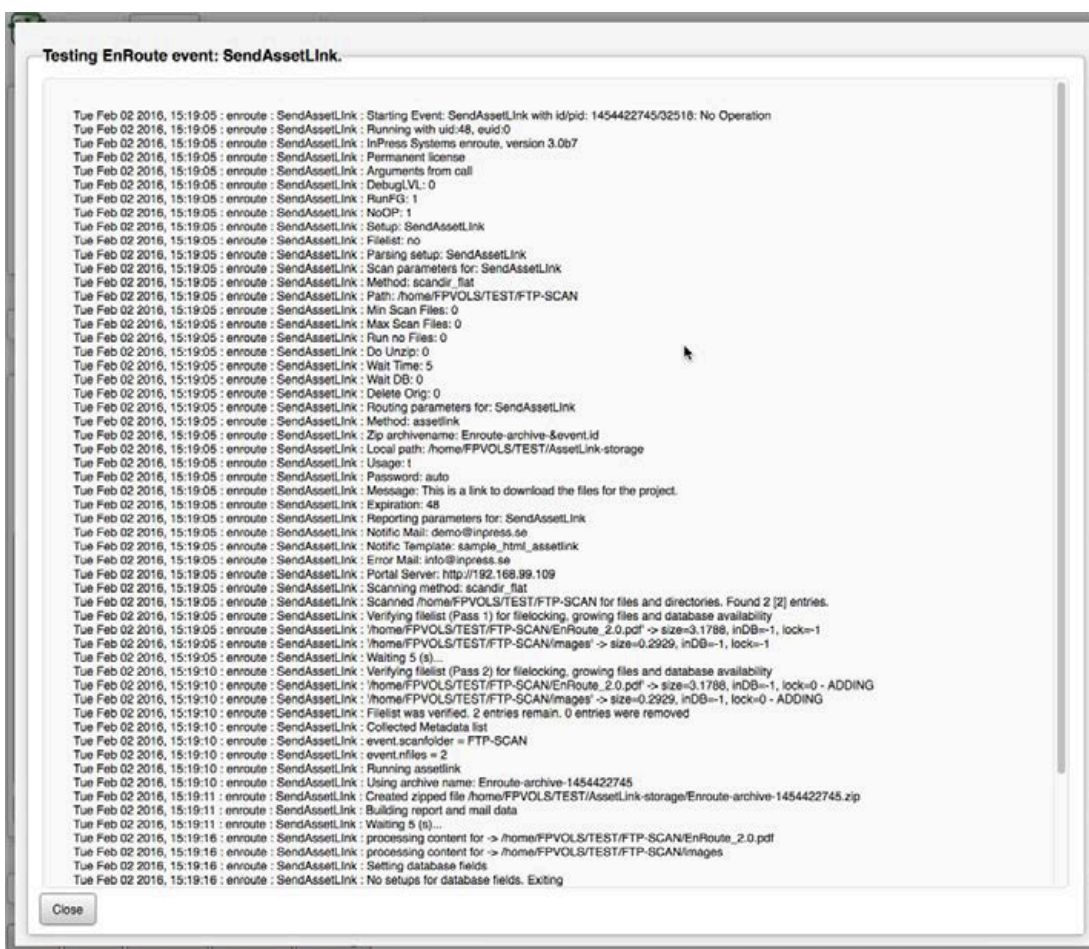
An event can be tested or executed from the admin GUI's event configuration page. Click the "test" button next to the save button to test an event. A new window will be open and the test result, including the files that are found, will be shown. No actual routing will take place.



Testing an event from the admin utility is easy...



It can take a while since event may wait for stable files etc...

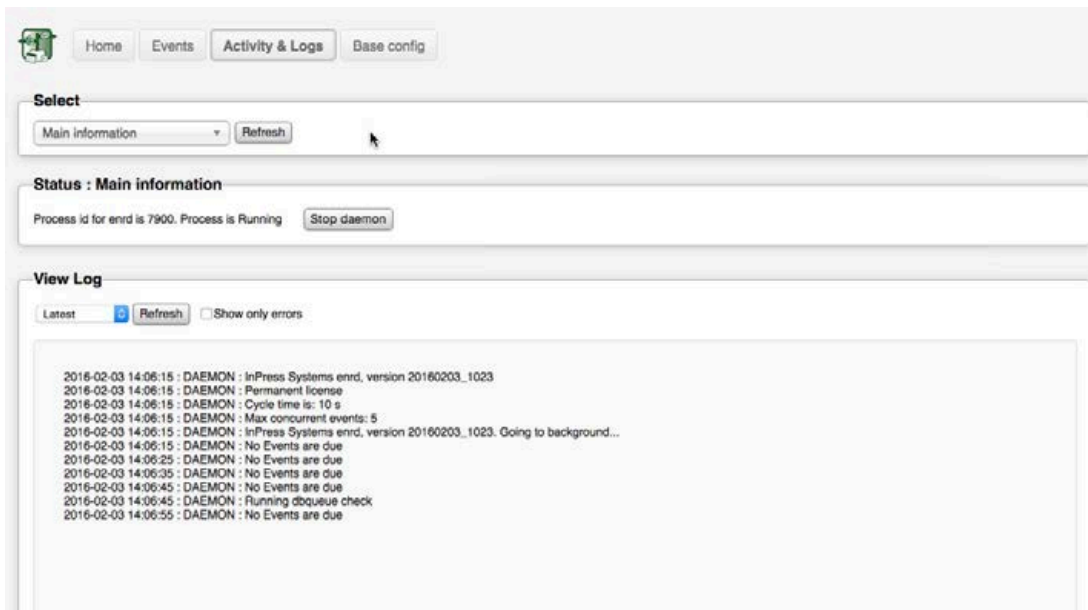


The full log is shown at the end

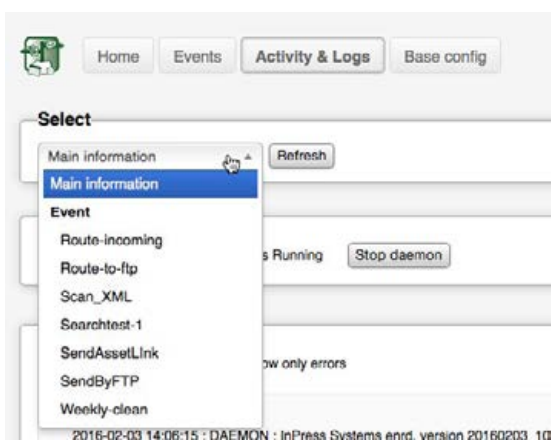
Clicking "Run Event" will fully execute the event, including running the route method.

5 ACTIVITY AND LOGS

The Activity and logs page contains main logs as well as individual Event logs. Select the log to view from selector at the top. It is also possible to jump to the log of an individual event from the events list and event configuration pages.




Main log



Select log to view

The top section show the current status of the event or the main daemon with ability stop the daemon.

For an event log, it is possible to select "details" or "download" to get the full (verbose) log for an event execution.


[Home](#)
[Events](#)
[Activity & Logs](#)
[Base config](#)

Select

SendAssetLink
Refresh

Status : Event : SendAssetLink

Last executed at: 2016-02-02 [Tue] 16:57:23

Next execution set to: -

View Log

Latest
Refresh
☐ Show only errors

```

2016-02-02 10:41:25 : 1454406085 : Scanned /home/FPVOLS/TEST/FTP-SCAN for files and directories. Found 2 [2] entries.
2016-02-02 10:41:30 : 1454406085 : Filelist was verified. 2 entries remain. 0 entries were removed
2016-02-02 10:41:30 : 1454406085 : Running assetlink
2016-02-02 10:41:30 : 1454406085 : Using archive name: Enroute-archive-1454406085
2016-02-02 10:41:31 : 1454406085 : Waiting 5 (s)...
2016-02-02 10:41:36 : 1454406085 : Setting database fields
2016-02-02 10:41:36 : 1454406085 : No setups for database fields. Exiting
2016-02-02 10:41:36 : 1454406085 : Use First email address: demo@inpress.se
2016-02-02 10:41:36 : 1454406085 : Ending Event: SendAssetLink with pid: 18030. Total execution time: 11 (s)
2016-02-02 10:41:36 : 1454406085 : Setting next execution to [0]

Event identifier 1454428614 View details Download
2016-02-02 16:56:54 : 1454428614 : Starting Event: SendAssetLink with id/pid: 1454428614/5608
2016-02-02 16:56:54 : 1454428614 : InPress Systems enroute, version 3.0b7
2016-02-02 16:56:54 : 1454428614 : Scanning method: scandir_flat
2016-02-02 16:56:54 : 1454428614 : Scanned /home/FPVOLS/TEST/FTP-SCAN for files and directories. Found 2 [2] entries.
2016-02-02 16:56:59 : 1454428614 : Filelist was verified. 2 entries remain. 0 entries were removed
2016-02-02 16:56:59 : 1454428614 : Running assetlink
2016-02-02 16:56:59 : 1454428614 : Using archive name: Enroute-archive-1454428614
2016-02-02 16:56:59 : 1454428614 : Waiting 5 (s)...
2016-02-02 16:57:04 : 1454428614 : Setting database fields
2016-02-02 16:57:04 : 1454428614 : No setups for database fields. Exiting
2016-02-02 16:57:04 : 1454428614 : Use First email address: demo@inpress.se
2016-02-02 16:57:04 : 1454428614 : Ending Event: SendAssetLink with pid: 5608. Total execution time: 10 (s)
2016-02-02 16:57:04 : 1454428614 : Setting next execution to [0]

```

Event log

View Log

Latest
Refresh
☐ Show only errors

```

Download
2016-02-02 16:56:54 : SendAssetLink : Starting Event: SendAssetLink with id/pid: 1454428614/5608
2016-02-02 16:56:54 : SendAssetLink : Running with uid:48, euid:0
2016-02-02 16:56:54 : SendAssetLink : InPress Systems enroute, version 3.0b7
2016-02-02 16:56:54 : SendAssetLink : Permanent license
2016-02-02 16:56:54 : SendAssetLink : Arguments from call
2016-02-02 16:56:54 : SendAssetLink : DebugLVL: 0
2016-02-02 16:56:54 : SendAssetLink : RunFG: 1
2016-02-02 16:56:54 : SendAssetLink : NoOP: 0
2016-02-02 16:56:54 : SendAssetLink : Setup: SendAssetLink
2016-02-02 16:56:54 : SendAssetLink : Filelist: no
2016-02-02 16:56:54 : SendAssetLink : Parsing setup: SendAssetLink
2016-02-02 16:56:54 : SendAssetLink : Scan parameters for: SendAssetLink
2016-02-02 16:56:54 : SendAssetLink : Method: scandir_flat
2016-02-02 16:56:54 : SendAssetLink : Path: /home/FPVOLS/TEST/FTP-SCAN
2016-02-02 16:56:54 : SendAssetLink : Min Scan Files: 0
2016-02-02 16:56:54 : SendAssetLink : Max Scan Files: 0
2016-02-02 16:56:54 : SendAssetLink : Run no Files: 0
2016-02-02 16:56:54 : SendAssetLink : Do Unzip: 0
2016-02-02 16:56:54 : SendAssetLink : Wait Time: 5
2016-02-02 16:56:54 : SendAssetLink : Wait DB: 0
2016-02-02 16:56:54 : SendAssetLink : Delete Orig: 0
2016-02-02 16:56:54 : SendAssetLink : Routing parameters for: SendAssetLink
2016-02-02 16:56:54 : SendAssetLink : Method: assetlink

```

Event log details of specific execution

A TIME AND DATE FORMATTING

When using the `date()` function as a part of a route path, reports or in other types of tasks, EnRoute uses the standard UNIX/C `time_t` type formatting of times and dates. As an example, using `date()` with the following formatter “%Y-%m-%d-%H.%M.%S”

```
date(%Y-%m-%d-%H.%M.%S)
```

Assuming that the current date is March 6, 2014 and the current time is 11.22.15, the output produced is

```
2014-03-06.11.22.15
```

Below is a partial list of sequences that can be used

%a	Locale's abbreviated weekday name
%A	Locale's full weekday name
%b	Locale's abbreviated month name
%B	Locale's full month name
%d	Day of month [1,31]; single digits are preceded by 0
%D	Date as %m/%d/%y
%e	Day of month [1,31]; single digits are preceded by a space
%h	Locale's abbreviated month name
%H	Hour (24-hour clock) [0,23]; single digits are preceded by 0
%I	Hour (12-hour clock) [1,12]; single digits are preceded by 0
%j	Day number of year [1,366]; single digits are preceded by 0
%k	Hour (24-hour clock) [0,23]; single digits are preceded by a blank
%l	Hour (12-hour clock) [1,12]; single digits are preceded by a blank
%m	Month number [1,12]; single digits are preceded by 0
%M	Minute [00,59]; leading 0 is permitted but not required
%p	Locale's equivalent of either a.m. or p.m
%r	Appropriate time representation in 12-hour clock format with %p
%R	Time as %H:%M
%S	Seconds [00,61]; the range of values is [00,61] rather than [00,59] to allow for the occasional leap second and even more occasional double leap second
%T	Time as %H:%M:%S
%u below	Weekday as a decimal number [1,7], with 1 representing Monday. See NOTES below
%U	Week number of year as a decimal number [00,53], with Sunday as the first day of week 1
%V	The ISO 8601 week number as a decimal number [01,53]. In the ISO 8601 week-based system, weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. See NOTES below
%w	Weekday as a decimal number [0,6], with 0 representing Sunday
%W day of week 1	Week number of year as a decimal number [00,53], with Monday as the first day of week 1
%x	Locale's appropriate date representation
%X	Locale's appropriate time representation
%y	Year within century [00,99]
%Y	Year, including the century (for example 1993)

InPress Systems Software License

This InPress Systems end user software license agreement ("agreement") is the legal agreement that governs your use of the software made available by InPress Systems AB (together with its accompanying documentation, the "software"). This agreement is between you, the customer who has acquired the software ("you"), and InPress Systems AB ("InPress Systems"). Please read this agreement carefully.

InPress Systems is only willing to provide the software to you on the condition that you accept all of the terms contained in this agreement. You accept this agreement by installing or using the software or installing a license for the software. By accepting this agreement or by installing the software, you represent and warrant that you have the authority to enter into this agreement, personally or if you have named a company as customer, on behalf of the company named as customer, and to bind either yourself or such company to the terms of this agreement.

If you did not acquire the software from InPress Systems or from an authorized InPress Systems integrator or a InPress Systems affiliate then you may not enter into this agreement or use the software. No other party has the right to transfer a copy of the software to you.

If you are unwilling to accept this agreement, do not use the software. If you have already paid for the software without having a prior opportunity to review this agreement and are now unwilling to agree to these terms, you may, within ten (10) days after the date on which you acquired the software, return it to InPress Systems or the authorized integrator from whom you acquired it, along with its original packaging and proof-of-purchase, for a full refund.

Notwithstanding anything herein to the contrary, no authorized InPress Systems integrator acts as an agent of InPress Systems, and no such party may enter into any contracts on behalf of InPress Systems. no authorized integrator has the authority to modify the terms of this agreement.

1. Grant of License

InPress Systems grants to you a nonexclusive, non-transferable license to use the Software on one computer system and to make one copy of the software solely for backup purposes. You must place the same copyright and other proprietary rights notices on any copy of the Software as appears on the original. You must not transfer, sell, assign, rent or distribute any copies of the Software to others. InPress Systems reserves all rights not expressly granted to you.

2. Proprietary Rights

As a licensee, you own the media on which the Software is originally recorded. The Software is copyrighted by and proprietary to InPress Systems and its suppliers. InPress Systems and its suppliers retain title and ownership of all copies of the Software. The nonexclusive license set forth in this Agreement is not a sale of the Software or any copy. You agree that you will not assign, sublicense, transfer, pledge, lease or share your rights under this Agreement and agree to take all reasonable steps to prevent unauthorized use. You agree you may not reverse assemble, reverse compile, or otherwise translate the software.

3. License Maintenance and Support

One year of support is included when purchasing InPress Systems products.

The year of support is calculated from the product licensing date or 30 days after the InPress Systems invoice is issued, whichever occurs first. Additional support is purchased at yearly intervals for 15% of the Current retail price of the software. (Please Note: Product modules that are added to the primary license after the original purchase, will be added to the existing support contract of the primary product license and therefore a full year of support will not be included in such cases).

4. No Other Rights

Except as stated above, this Agreement does not grant you any rights to patents, copyrights, trade secrets, trade names, trademarks (whether registered or unregistered), or any other rights, franchises, or license in respect of the Software. You MAY NOT MODIFY TRANSLATE, DISASSEMBLE, OR DECOMPILE THE SOFTWARE OR ANY COPY, IN WHOLE OR IN PART.

5. Term

The license is effective until terminated. You may terminate the license at any time by destroying the Software (including the related documentation) together with all copies or modifications in any form. InPress Systems will have the right to terminate your license immediately if you fail to comply with any term or condition of the Agreement. Upon any termination you must destroy the Software together with all copies or modifications in any form.

6. LIMITED WARRANTY

6.1 InPress Systems warrants to you that the Software will perform substantially in accordance with the user's manual for a period of thirty (30) days after delivery to you ("Warranty Period"). If the Software fails to comply with this limited warranty, InPress Systems will at its option and at no cost to you, correct errors you discover which you report during the Warranty Period, or replace the Software, or refund the license fee paid for the Software provided you return the Software.

6.2 INPRESS SYSTEMS AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. YOU UNDERSTAND THAT, EXCEPT FOR THE EXPRESS WARRANTY SET FORTH IN SECTION 6.1, INPRESS SYSTEMS AND ITS SUPPLIERS MAKE NO WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, OR STATUTORY, WITH RESPECT TO THE SOFTWARE, INCLUDING ANY WARRANTIES AS TO PERFORMANCE, NON-INFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. EXCEPT FOR THE EXPRESS WARRANTY STATED IN SECTION 6.1, THE SOFTWARE IS PROVIDED "AS IS" WITH ALL FAULTS, AND THE ENTIRE RISK AS TO SATISFACTORY QUALITY, ACCURACY, AND EFFORT IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY INPRESS SYSTEMS OR ANY AUTHORIZED INTEGRATOR, AGENTS OR EMPLOYEES.

7. LIMIT OF LIABILITY

IN NO EVENT WILL INPRESS SYSTEMS OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, OR DAMAGES FOR ANY LOST DATA OR LOST PROFITS, ARISING FROM OR RELATING TO THIS AGREEMENT, EVEN IF INPRESS SYSTEMS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. INPRESS SYSTEMS' TOTAL CUMULATIVE LIABILITY IN CONNECTION WITH THIS AGREEMENT AND THE SOFTWARE, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL NOT EXCEED THE AMOUNT OF LICENSE FEES PAID TO INPRESS SYSTEMS OR YOUR AUTHORIZED INTEGRATOR, AS APPLICABLE, HEREUNDER. YOU ACKNOWLEDGE THAT THE LICENSE FEES REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT INPRESS SYSTEMS WOULD NOT ENTER INTO THIS AGREEMENT WITHOUT THESE LIMITATIONS ON ITS LIABILITY. IN ADDITION, INPRESS SYSTEMS DISCLAIMS ALL LIABILITY OF ANY KIND OF INPRESS SYSTEMS' SUPPLIERS.

8. Integration.

You acknowledge that you have read this Agreement, understand it, and that by installing the software you agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement of the agreement between InPress Systems and you which supersedes any proposal or prior agreement, oral or written, and any other communications between InPress Systems and you relating to the subject matter of this Agreement. No variation of the terms of the Agreement or any different terms will be enforceable against InPress Systems unless InPress Systems gives its express consent, including an express waiver of the terms of this Agreement, in writing signed by an officer of InPress Systems.

9. Governing Law

This Agreement shall be governed by and construed in accordance with the laws of Sweden without giving effect to the choice of law principles thereof.

10. Arbitration

Any dispute, controversy or claim arising out of or in connection with this Agreement, or the breach, termination or invalidity thereof, shall be settled by arbitration in accordance with Göteborgsklausulerna om skiljeförfarande (simplified rules of arbitration). The arbitral tribunal shall be composed of one arbitrator.

InPress Systems Software Credits

InPress Systems AB use the commonly available software libraries listed below.

jQuery

<http://jquery.com>

jQuery is provided under the MIT license.

jQuery UI

<http://jqueryui.com>

Dual licensed under the MIT or GPL Version 2 licenses.

TipTip

<http://code.drewwilson.com/entry/tiptip-jquery-plugin>

This TipTip jQuery plug-in is dual licensed under the MIT and GPL licenses.

JQZoom

<http://www.mind-projects.it/projects/jqzoom/>

This software is licensed under BSD.(read the license inside the archive)

jCrop

<http://deepliquid.com/content/Jcrop.html>

Jcrop is free software released under MIT License.

Fancybox

<http://fancybox.net>

Licensed under both MIT and GPL licenses

Chosen

<http://harvesthq.github.io/chosen/>

Chosen is licensed under the MIT license.

libcurl

<http://curl.haxx.se/docs/copyright.html>

Curl and libcurl are licensed under a MIT/X derivate license

libsqlite

<http://www.sqlite.org/copyright.html>

Public domain

libmysqlclient

GNU General Public License

libqrencode

GNU Lesser General Public License