



AGETOR®

DocDeliver v. 2.0.10
what's new?

Content

1	Document abstract.....	2
2	Installation.....	3
2.1	Upgrade notes.....	3
3	Configuration files – new structure.....	3
4	Multithreaded processing in v. 2.0.10.....	4
4.1	Inherent problems with single threaded communication to multiple peers.....	4
5	Processing and “inactivity”.....	4
6	Notification.....	5
6.1	Job delivery errors.....	5
6.2	Notification on inactive deliveries.....	5
6.3	Notification on many undelivered jobs.....	5
7	Common delivery properties.....	6
8	New media types and properties.....	7
8.1.1	AXT media.....	7
9	Media configuration.....	7
9.1	Media configuration format.....	8
9.2	Media connection properties.....	8
9.3	Base properties all media.....	9
9.4	Base media configurations (plugged-in media).....	10
9.5	Destination specific properties.....	10
10	Job property resolution.....	10
11	Delivery process.....	11
11.1	Delivery queues.....	11
11.2	Delivery restrictions.....	12
11.3	Connection reuse.....	12
11.3.1	Concept.....	12
11.3.2	Controlling reuse.....	13
11.4	Delivery retries.....	13
12	Logical servers.....	13
13	Obtaining status information.....	13
13.1	General status information.....	14
13.2	Currently receiving.....	15
13.3	Delivering.....	15
13.4	Queue summaries.....	15
13.5	Media client pools.....	15

1 Document abstract

This document describes the changes of the AXT DocDeliver Service (the DDS) implemented in v. 2.0.10. For full documentation of the DDS, or if you are not already familiar with the DDS delivery process, please refer to the document AXT DocDeliver User Guide.

The major focus for this release has been to improve the robustness in face of misbehaving media (FTP-servers and other communication maladies). The previous versions of the DDS used a single-threaded architecture for fetching and processing files which made it vulnerable to such circumstances. This version supports multiple processors for delivery.

The main improvements are:

- Configurable number of processors increasing possible throughput and QOS
- Automatic takeover by new processors if processor becomes inactive or *locked* by badly behaving communication peer (e.g. FTP-server)
- Surveillance of processors to ensure their healthiness
- Configurable mail-notification in case of processor timeout (inactivity) or many undelivered files
- Improved status information in the AGETOR® Control Center
- Possible per-destination configurations of communication and delivery properties (retries, retry intervals, timeouts, max simultaneous connections etc.)
- AXT media plug-in, allowing redelivery of data to AXT transformations (use with latest versions of AXT FTPInlet and AXT MailInlet).

2 Installation

The DDS runs with AXT Basic v.2.0.4, but it is recommended to upgrade to latest version first. Also you may take full advantage of the new AXT delivery media by upgrading your FTP-inlet and Mail-inlet since these will store files not deliverable in DDS-format for possible redelivery to AXT.



Note that this version supplies media default properties in files located in directory `${AGETOR_HOME}/conf/docdeliver/mediacnf`. It is not recommended to modify these files but values may be overridden in more specific configuration files (see section on delivery configuration).

2.1 Upgrade notes



This version is backward compatible with existing configuration files and comes with sensible defaults for configurable properties. However, to take full advantage of the new functionality it is recommended to review the configuration and add properties such as a notification mail-address.



The factory tags of the main configuration file are no longer necessary (and may be removed) since factory-tags are now placed in media specific configuration files.



This version of the DDS is *not directly compatible with media plug-ins compiled for previous versions!* The plug-in interfaces have been changed slightly and proprietary media plug-ins require a minor rewrite/recompile.



The format of information files (.xml) used to store job properties in have been changed. This means that old failed jobs should be delivered before you upgrade to this version.

3 Configuration files – new structure

Besides the main configuration file used by the DDS, one or more configuration files are now present in directory `${AGETOR_HOME}/conf/docdeliver/mediacnf`. These files holds default properties for each plugged in media.

There is full compatibility with existing configuration files since these extra configuration files are read independently of the main configuration. These new files simply add more flexibility to the way submitted jobs are enriched with default properties depending on their destination.

4 Multithreaded processing in v. 2.0.10

4.1 Inherent problems with single threaded communication to multiple peers

Since the DDS may communicate with a large number of remote servers of varying stability through unreliable networks with multiple intervening servers (Firewalls, routers etc.), the communication with remote servers may be subject to errors such as broken connections, extremely slow data transfer or in worst case no data transfer at all for longer or shorter periods.

Under such circumstances, single threaded processing will act as a bottleneck to the total processing since all files must await the termination of processing of the error prone server/file. This could be perceived as if the DDS was inactive or *hanging* for periods.

This version of the DDS supports multithreaded processing; this model has a number of advantages – e.g. better robustness to errors and higher possible throughput. Better robustness results from the fact that while a slow or halted transfer is in progress, other transfers may take place. Likewise the transfer of e.g. a large file from a slow server can take place while multiple smaller files are fetched and transformed simultaneously using other processing threads.

5 Processing and “inactivity”

Since the DDS may communicate with a large number of remote servers of varying stability through unreliable networks with multiple intervening servers (Firewalls, routers etc.), the communication with remote servers may be subject to errors such as broken connections, extremely slow data transfer or in worst case no data transfer at all for longer or shorter periods.

DDS supports multithreaded processing; this model has a number of advantages – e.g. better robustness to errors and higher possible throughput. Better robustness results from the fact that while a slow or halted transfer is in progress, other transfers may take place. Likewise the transfer of e.g. a large file from a slow server can take place while multiple smaller files are fetched and transformed simultaneously using other processing threads.

Extremely slow communication may not necessarily mean that a process is in error but simply that it takes a long time to complete. However, it is perceived as a halt in the overall processing since the thread is bound for a long time. To detect such scenarios we define the maximum acceptable processing time for an *active* process and perceive processes that take longer time as “inactive”. We quote the word since the thread may not really be absolutely inactive and *may* complete its task in time. Once a process becomes inactive the number of active processes drops and we allow the creation of a new active process. In this way the pool of active processes may stay constant while we don’t stop slow transfers that may eventually complete.

In the main configuration we specify the number of active and inactive processes with the `maxActiveDeliveryThreads` and `maxInactiveDeliveryThreads` properties:

```
<docdeliver
. . .
maxActiveDeliveryThreads = "4"
maxInactiveDeliveryThreads = "4"
```



Note that the chosen number of processes is to be shared among all deliveries. The above settings will for instance allow two FTP deliveries, one Mail delivery and one file system storage to take place at the same time (totalling four active deliveries).

A word of caution; the more processor threads that are used, the more simultaneous processing will take place. This may result in heavy load of the target machine if it is required to transform multiple complex files at the same time. Thus you may want a limited number of processor threads e.g. 2-5 depending on typical file sizes (small files may be better candidates for multitasking than big), the number of potential destinations etc.

6 Notification

The DDS supports mail-notification in case of delivery or processing problems. To set up a receiver of notifications adjust the following attributes of the main configuration tag attributes shown:

```
<docdeliver
. . .
smtpserver = "mailserver.acme.com"
errorFrom = "docdeliver@acme.com"
errorTo = "admin@acme.com"
errorSubject = "DocDeliver error" ...
```

Once defined, the mail user denoted by the `to` attribute will receive a mail in the following situations:

- On DDS start-up
- On job delivery errors (no connection, bad login etc.)
- When an active thread becomes inactive (warning of potential problematic situation)
- When the maximum number of undelivered files is reached (warning of potential problematic situation)
- When the number of undelivered files fall below the critical point again (cancellation of problematic situation)

The situations are detailed below. For information about error resolution please refer to the DocDeliver User Guide.

6.1 Job delivery errors

This works as usual; when a document cannot be delivered an error mail is sent with a specification of the job-id and the error in short.

6.2 Notification on inactive deliveries

A delivery process going inactive is not necessarily a severe problem (since it may complete just fine in time) but it is something we may want to be aware of. The mail will detail the current state of the DDS in a fashion similar to the view you may obtain in the AGETOR® Control Center (ACC). Please refer section 13 for details on status information.

6.3 Notification on many undelivered jobs

Jobs for delivery are received in the delivery directory and delivered from here. In some cases the delivery processes may not be able to keep up with the rate at which jobs are received. This may be caused by heavy batch deliveries from multiple clients or when the machine on which the DDS is running is overloaded. Often this bulking up of documents for delivery is temporary and no problem.

However, in some installations it is critical that documents are delivered timely and delays in delivery must be known. To this end the DDS allows you to specify the maximum number of jobs in the delivery directory. When this value is exceeded a notification mail is sent to the configured receiver so appropriate action can be taken. Since the problem may be temporary the DDS will also send a cancellation mail if the problem is resolved by itself (i.e. the number of undelivered files drops significantly again).

The max number of files is specified as an attribute to the `docdeliver` root tag of the main configuration file:

```
<docdeliver
.
.
.
undeliveredFilesWarningLimit = "100"
```

Example 1: specifying the limit for undelivered files (causing a mail notification)

7 Common delivery properties

The list below contains all properties common to all media. I.e. they can be used in connection with any delivery since they are not media specific. New properties are **boldfaced**.

Parameter	Type	Description	Defaults (with order)
<code>media</code>	Required		
<code>logicalServer</code>	Optional	This parameter is the key to a logical server that is configured in the DDS configuration. Please refer to section 12 on logical server definitions.	<i>none</i>
<code>output</code>	Optional	Whether or not the input to the filter should be returned as output from the filter. If true the document is returned. If false, the id of the delivery is returned as XML from DDS. This parameter is deprecated since AXT provides a standard way of doing this; use the <code>output="input"</code> attribute of the filter tag to do this.	<i>false</i>
<code>tentatives</code>	Optional	How many times the service should try to connect to the server. When the number of attempts reaches this value the job will be moved to the failed folder and the DDS will give up on it.	3
<code>retryInterval</code>	Optional	The number of seconds between retries made.	30
<code>savedata</code>	Optional	If set to <code>true</code> the DDS will save a copy of the document in the succeed folder together with the job information. When not set the document will be deleted after it has been transferred successfully to the destination.	<i>false</i>
<code>synchronized</code>	Optional	If set to <code>true</code> the filter will wait for server to finish job, before it continues. Note that if server needs to retry several times before connecting, timeout to server might occur before server has finished the job. Increasing <code>servertimeout</code> alleviates this issue.	<i>false</i>
<code>priority</code>	Optional	Denotes the job priority. When multiple jobs are due for delivery, jobs with a higher priority get precedence.	10

queuingkey	Optional	The queuing key determines what delivery queue the job goes to. Jobs within the same queue are delivered sequentially, but jobs in different queues may be delivered in parallel. Normally you won't need to override this property.	<i>Configurable per media in DDS</i>
deliveryGroupingKey	Optional	This key determines what client connection pool the job maps to. The key is configured per media on the server and constructed from media specific properties that ensure that jobs mapping to the same key are to the same destination (server, path, user-wise). This allows reuse of open connection channels.	<i>Configurable per media in DDS</i>

Table 1: Parameters common to all media

8 New media types and properties

The AXT Media have been added. The primary reason is that you may now redeliver files fetched by the AXT FTPInlet or AXT MailInlet since these now store such files in DDS compatible format (a .xml and a .dat file).

8.1.1 AXT media

When the value of the "media" parameter is set to "axt" the data will be sent to AXT for transformation.

8.1.1.1 Media specific properties



All the common parameters can be used together with these:

Property	Type	Description	Defaults (with order)
axturl	Optional	The URL of the AXT server to send data to	http://localhost/servlet/dk.bording.axt.server.ContainerServlet
__axtkey__<keyname>	Optional	One or more AXT keys may be given using this syntax where the prefix __axtkey__ is removed before the <keyname> part is sent with the document for processing in the AXT server	

The AXT media is a responding media type and a successful invocation will result in a file named Job_res_<job-id>.dat in the succeed directory.

9 Media configuration

With version 2.0.10 fine-grained control of media and destination specific properties have been added. These include connection properties (timeout, max connections to a specific destination, connection reuse etc.) as well as delivery specific properties such as what jobs are delivered in strict sequence, job priority; numbers of retries and retry intervals.

9.1 Media configuration format

Media configuration may be placed either directly in the `docdeliver.xml` configuration file or in separate files below the directory `$AGETOR_HOME/conf/docdeliver/mediaconf`. The media configurations take the form (whether inlined in main configuration or in a separate file):

```
<mediaConfiguration>
  <confPattern media="*" server="*">
    <properties .../>
  </confPattern>
</mediaConfiguration>
```

The `confPattern` tag denotes what media and what destination server the configuration applies to. The two attributes of this tag (media and server) may take on specific values such as `"ftp"` + `"ftpserver.acme.com"` or the generic specification `"*"`. This means that you may have configurations applying to all media/destinations; a concrete media and all destinations; all media and a specific destination; and finally a concrete media and a concrete destination (see examples below).

```
<mediaConfiguration>
  <confPattern media="*" server="*">
    <properties .../>
  </confPattern>
</mediaConfiguration>
```

Example 2: media configuration applying to all media and servers

```
<mediaConfiguration>
  <confPattern media="ftp" server="*">
    <properties .../>
  </confPattern>
</mediaConfiguration>
```

Example 3: media configuration applying to all FTP deliveries regardless of destination server

```
<mediaConfiguration>
  <confPattern media="mail" server="ms.acme.com">
    <properties .../>
  </confPattern>
</mediaConfiguration>
```

Example 4: media configuration applying *only* to mail deliveries to the mail server at `ms.acme.com`

The interesting part, of course, is the property tag, which contains the actual settings for the media/destination specification. Here any property that is valid for the media in question may be given. Obviously this includes properties applying to all media.



For configurations matching all media it is important to note that *only* common properties may be given. I.e. properties in Example 2 above cannot refer to properties for FTP, Mail or other concrete media.

9.2 Media connection properties

The following properties are only used in media configuration files (i.e. not from the outlet filter or with logical servers) and state properties for the communication with the specific destination.

Property	Type	Description	Defaults
----------	------	-------------	----------

MaxConnections	Optional	Denotes the max number of simultaneous clients to the destination.	2
MaxDeliveriesOpenConnection	Optional	Specifies how many documents may be sent on a connection once it has been opened. Some media will allow any number, whereas other media by nature allows only one delivery. For media allowing multiple deliveries this property allows tuning to the destination; reusing the open connection is more efficient than connecting and logging in multiple times.	1
MaxDeliveryTimeSec	Optional	The number of seconds a delivery may take before being perceived as inactive (see section Fejl! Henvisningskilde ikke fundet.).	1800
PooledConnectionMaxIdleAgeSec	Optional	The number of seconds a connection may be open and idle before a new connection is created. Since the destination server may choose to close connections that are not used for a period, we want to specify a max time that a connection is valid with no activity over it.	45

9.3 Base properties all media

The DDS is shipped with a base media configuration file named `defaultmedia.xml`. This file uses the generic `*`-pattern for server and media to match all media. This is the base property file from which all more specific media configurations inherit their properties.

```
<?xml version="1.0" ?>
<mediaConfiguration>
  <!--
    For this generic pattern only generic attributes have effect. They will apply to all
    more specific definitions unless these override the attributes.
  -->
  <confPattern media="*" server="*">
    <properties
      maxConnections="2"
      maxDeliveriesOpenConnection="1"
      maxDeliveryTimeSec="1800"
      pooledConnectionMaxIdleAgeSec="45"
      timeout="30"
      tentatives="3"
      retryInterval="60"
      priority="5"
      savedata="false"
      synchronized="false"
      queuingkey="{media}"
      deliveryGroupingKey="{media}"
    />
  </confPattern>
</mediaConfiguration>
```

Example 5: Basic media properties in `defaultmedia.xml`

Concrete media like the FTP, Mail and AXT media must provide their own configurations as well that specifies the *media specific* properties. Also these configurations may want to *override* properties from the default media configuration file.

Properties not overridden are inherited.

9.4 Base media configurations (plugged-in media)

By default a number of media are supported by the DDS. These include FTP, Mail, File system and AXT deliveries. Any plugged in media must have a configuration file specifying the defaults for the media specific properties as well as media specific settings for general properties. The corresponding configurations may be found in the `mediaconf` directory. These base configurations have the general pattern `*` for the server attributes to cover all destinations for the media.

The base configurations must also have the `mediaFactory` tag specifying the factory class for the media.

```
<mediafactory factory = "dk.bording.axt.docdeliver.media.ftp.FTPMediaFactory" />
```

In this manner the base configurations determine if the media is loaded and with what base properties. Thus it is simple to add new media types and subsequently adjust/override their properties.

9.5 Destination specific properties

Yet another level in the property inheritance hierarchy may be defined to achieve destination specific properties.

If you wish to set specific properties for a certain destination this could be achieved as in the example below for the FTP-server `ftpserver.acme.com`. In this definition we override values for properties `maxConnections` to allow up to five deliveries at the same time to this destination. The `fileparser` attribute ensures that the communication with the FTP-server takes into account that it uses a special format (Linux). We also specify that we should retry deliveries five times to this server with an interval of 60 seconds. Finally, we instruct the DDS that data files should be kept even when transfer succeeds (`savadata`).

```
<!--
  Example server specifying a special fileparser and connection properties.
-->
<confPattern media="ftp" server="ftpserver.acme.com">
  <properties
    maxConnections="5"
    fileparser="dk.bording.axt.client.ftp.LinuxFTPFileListParser"
    tentatives="5"
    retryInterval="60"
    savadata="true"
  />
</confPattern>
```

All other properties are inherited from base configurations or overridden by received properties for the job.

10 Job property resolution

Jobs are received by the DDS through its server interface. At present jobs are primarily received from AXT through the `DocDeliverOutlet` filter. With this filter you may specify some or all

general as well as media specific properties for the job. However, most of the time it is more convenient only to pass the strictly necessary properties and have the DDS resolve the missing values from configuration. The DDS uses the media type and destination properties to look-up the best match for the job using the following scheme:

- (i) First the job is enriched with missing properties valid for all media and destinations (media="*" server="*")
- (ii) Then properties for the media in question are used (e.g. properties from the basic ftp configuration)
- (iii) Then properties for the specific destination (if a definition is present) are used
- (iv) If a logical media key was specified for the job, the logical definition is looked up and applied thus overriding any properties from the previous steps
- (v) Finally the properties received with the job override properties with same name from previous steps.

It is important to understand the precedence of properties – i.e. that job properties > logical server properties > configured destination properties > media properties > general properties.

The used scheme has the following advantages:

- Minimal job specification from client (media and delivery information is in server)
- Fallback defaults for all jobs whether they are to known destinations or not
- Option of fine-tuning properties of specific media and destinations
- Easy plug-in of new media types with sensible yet tuneable defaults

11 Delivery process

This section describes the details of the delivery process. If you don't care for the details you can skip the section. Unless you are writing a DDS plug-in or wish to fine-tune delivery properties the section may not be relevant to you.

11.1 Delivery queues

Once a job is received and stored in the delivery directory, it is put in an internal delivery queue. The actual mapping of a job to a queue is media specific since the idea behind queues is to deliver related jobs sequentially. The notion of relation is most often that jobs are related and should be delivered sequentially (i.e. in the order they were received by the DDS) if they have the same destination. Thus the queue should be the same for jobs to the same destination. Since the specification of a unique destination is media specific (for FTP, a destination is uniquely specified by server, port and username whereas an AXT server is specified by URL (same URL equals same destination)).

By configuration for each media type the queue mapping is defined in terms of media attributes. Thus the queuing key for the FTP-media is specified as

```
queuingkey="{media}:{ftpserver}:{port}:{user}"
```

For the file media the queuing key is simply

```
queuingkey="{media}"
```

denoting that all files are mapped to the same queue and thus delivered sequentially to disk.

However, this approach allows for more advanced delivery rules that deliver jobs sequentially even if they are to different destinations or media. Clients may control this behaviour by providing the queuing key for the job and use the same keys for related jobs.

From AXT this could simply be expressed using the `queuingkey` attribute:

```
<filter class="dk.bording.axt.client.docdeliver.DocDeliverOutlet" >
  <param name="media" value="ftp"/>
  <param name="ftpserver" value="ftp.partner.com"/>
  <param name="folder" value="orders"/>
  <param name="mask" value="order%s.txt"/>
  <param name="serie" value="partner4_order"/>
  <param name="priority" value="20"/>
  <param name="queuingkey" value="partner4"/>
  <param name="user" value="p4"/>
  <param name="password" value="p4secret"/>
</filter>
```

11.2 Delivery restrictions

Delivery is restricted by a number of factors:

- 1) The global number of processing threads available
- 2) The order of jobs in queues (arrival time and queue mapping)
- 3) Job priority
- 4) Delivery restrictions pertaining to media/destinations

The two first factors have been covered in previous sections; if all processors are busy delivering data obviously an otherwise ready job in the head of a delivery queue has to wait its turn.

Jobs may be assigned a priority that specifies the urgency of the particular job. This priority does not affect the sequential delivery within a queue since the primary rule here is to deliver related jobs in time order. However when more jobs are ready for delivery from multiple queues, these are processed such that high priority jobs have precedence. By default all jobs inherit the default priority stated in the default media configuration file. As with all properties the priority may be overridden for specific media and destinations (see section 10 on property resolution).

The fourth restriction above pertains to properties that may be configured on media/destination level. For a certain media/destination you may specify the property `maxConnections`. This property states how many simultaneous deliveries that are allowed for the destination. E.g. a given FTP-server may be restricted to at most three clients at the same time or if a certain server is known to be particular capable (and you expect much traffic to it) you may specify a larger than default number of connections to the server.

11.3 Connection reuse

11.3.1 Concept

Over a time period a number of jobs headed for the same destination may arrive to the DDS. To make delivery efficient such jobs should be able to reuse the same connection for delivery if the underlying media support connection reuse.

Jobs that map to the same *delivery group* may be delivered using the same connection per definition. Thus, for the first job in such a sequence a connection is established using the properties of the job (e.g. server name, user, password etc. relevant for the media). Subsequent jobs mapping to the same group are simply transferred over the already open connection. As an implication of this

it is extremely important that all the jobs in the group actually have the same values for the attributes determining the channel. I.e. the key that maps to a delivery group should include *all* the attributes used for establishing the unique connection. For an FTP-connection the attributes `ftpserver`, `port` and `user` may be considered corresponding to a unique channel. All jobs with same values for these attributes should be sent through that channel. To completely disambiguate channels it is necessary to include the media in the key as well. For the FTP-media the following delivery queue key is used:

```
deliveryGroupingKey="{media}:{ftpserver}:{port}:{user}"
```

It is the responsibility of the media plug-in writer to provide correct delivery grouping key definitions since it is the plug-in writer that creates the connections to the media and knows what attributes are relevant.

11.3.2 Controlling reuse

Though the grouping key should not be modified, some channel reuse properties may be adjusted for a specific destination. The connection properties `maxDeliveriesOpenConnection` and `pooledConnectionMaxIdleAgeSec` control how many deliveries may be made over an open connection before the connection is re-established and how long time a connection may be idle before a new connection is made (see also section 9.2).

11.4 Delivery retries

When an error occurs during the delivery of a job, it may be fatal or not. If the DDS concludes that it may be a temporary problem (e.g. no connection to a server could be caused by temporary network problems), the job is retried the number of times specified by the `tentatives` attribute. Retries are performed with the interval specified by `retryInterval`.

When all retries have been performed without success the file is considered failed and moved to the fail folder. If a client is waiting for response (using the `synchronized` property for the job) the error is propagated back now.

12 Logical servers

The concept of logical servers has more or less lost its justification with the support for basic media configuration defaults, destination specific configuration and property inheritance that comes with version 2.0.10.



However the concept is still supported for compatibility.

13 Obtaining status information

From the AGETOR® Control Center (ACC) you may obtain status information on the DDS.

The view below illustrates the basic information displayed in the status view.

Status										
status: Server state : RUNNING Queued requests : 0										
General:										
Version	Started	State	Total jobs	Succeeded	Failed	On disk	Sleep time	Active	Inactive	
2.0.7	Tue Aug 16 13:46:48 CEST 2005	Running	4912	4909	0	4	30s	2/4	0/2	
Currently receiving files:										
Id	Media	Server	Received bytes	Inactive time						
156e5ed_105be901404_-5694	file	localhost	0	20						
Delivering:										
JobID	Date	Media	Server	Priority	Status					
156e5ed_105be901404_-5698	2005.08.16 14:17:32	ftp	nixon	10	Trying to upload the file					
156e5ed_105be901404_-5696	2005.08.16 14:17:32	mail	bdmail.bording.dk	10	Sending the mail (delivered 528 of 528 bytes)					
Queue summary:										
Queue id	type	Documents	Avg. doc time in queue	Total seen docs						
mail:bdmail.bording.dk:25:anonymous	strict_order	1	0.541s	814						
ftp:nixon:21:kursus	strict_order	1	0.481s	821						
axt:http://godzilla.8111/servlet/dk.bording.axt.server.ContainerServlet	strict_order	0	0.184s	825						
file	strict_order	0	0.39s	810						
FTP_QUEUE_NIX	strict_order	1	0.655s	818						
axt:http://localhost/servlet/dk.bording.axt.server.ContainerServlet	strict_order	0	0.223s	824						
Media client pools:										
Delivery key	Max conn.	Free/Active/Inactive	Tot. req.	Tot. created	Connects	Logins	Errors	Avg.use		
mail:bdmail.bording.dk:25:anonymous:0	1	0/1/0	814	83	83	83	0	0.366s		
axt:http://godzilla.8111/servlet/dk.bording.axt.server.ContainerServlet:0	5	1/0/0	825	825	825	825	0	0.152s		
file:4	5	1/0/0	810	41	41	41	0	0.29s		
ftp:nixon:21:kursus:1	5	1/1/0	1638	84	84	84	0	0.322s		
axt:http://localhost/servlet/dk.bording.axt.server.ContainerServlet:0	5	1/0/0	824	824	824	824	0	0.188s		

The information is organized as a number of table structures each detailing certain aspects of the current or historic state. The tables are described in subsections below.

13.1 General status information

This table has general information about the Inlet in general including version, start-up time, jobs processed etc. Here you can also see the current settings for threads (active/inactive process threads).

Column	Description
version	Shows the DDS version number
Started	The time that the DDS was started
State	The current state of the DDS
Succeeded	The total number of jobs successfully delivered since start-up
Failed	The total number of jobs failed since start-up
On disk	The number of jobs in the delivering directory not yet delivered. Note that the number reflect the number of files present at the last directory scan and it may thus be slightly different from the current number of files (files may have been delivered since the scan).
Sleep time	The value of the sleep attribute for the DDS. This specifies the frequency at which the delivery directory is scanned for new files.

Active	The active column shows the current number of active deliveries and the max number of simultaneous inactive deliveries allowed on the form <current>/<max>
Inactive	The inactive column shows the current number of inactive deliveries and the max number of simultaneous deliveries allowed on the form <current>/<max>

13.2 Currently receiving

This column describes jobs that are currently being received by the DDS.

Column	Description
Id	The job id
Media	Media type
Server	Destination server
Received bytes	Number of bytes received so far
Inative time	Number of milliseconds that no data was received

13.3 Delivering

Column	Description
JobId	The job id
Date	The time the job was received
Media	Destination media
Server	Destination server
Priority	Priority of job
Status	Last registered action for the delivery

13.4 Queue summaries

Describes the delivery queues.

Column	Description
Queue id	This is the queuing key for the queue
type	Denotes if the queue is read in strict order (sequentially). This is the only mode at present.
Documents	The number of documents in the queue ready for or undergoing delivery.
Avrg. doc time in queue	The average time a document has stayed in the queue before delivered or failed. This value says something about the delivery efficiency.
Total seen documents	The number of documents that have passed the queue in its lifetime.

13.5 Media client pools

Describes the state of media client pools. These are constructed from media configurations and have properties for max connections etc.

Column	Description
Delivery key	This is the delivery grouping key (see section 11.3)
max conn.	The configured maximum number of connections to this destination
free/active/inactive	free/active/inactive connections to the destination.

Tot.req.	The total number of jobs that mapped to this channel
Tot. created	The total number of media clients created in this connection pool
Connects	The total number of connections created in this connection pool
Logins	The total number of login requests made to this connection pool
Errors	The total number of errors on connection pool
Avg.use.	The average time a connection lasted (i.e. delivery through channel)