StreamServe™ Persuasion™

StreamServe Persuasion SP5 Connectors

User Guide

Rev B

StreamServe Persuasion SP5 Connectors User Guide
Rev B
© 2001-2010 STREAMSERVE, INC.
ALL RIGHTS RESERVED
United States patent #7,127,520

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of StreamServe, Inc. Information in this document is subject to change without notice. StreamServe Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this book. All registered names, product names and trademarks of other companies mentioned in this documentation are used for identification purposes only and are acknowledged as property of the respective company. Companies, names and data used in examples in this document are fictitious unless otherwise noted.

StreamServe, Inc. offers no guarantees and assumes no responsibility or liability of any type with respect to third party products and services, including any liability resulting from incompatibility between the third party products and services and the products and services offered by StreamServe, Inc. By using StreamServe and the third party products mentioned in this document, you agree that you will not hold StreamServe, Inc. responsible or liable with respect to the third party products and services or seek to do so.

The trademarks, logos, and service marks in this document are the property of StreamServe, Inc. or other third parties. You are not permitted to use the marks without the prior written consent of StreamServe, Inc. or the third party that owns the marks.

Use of the StreamServe product with third party products not mentioned in this document is entirely at your own risk, also as regards the StreamServe products.

StreamServe Web Site
http://www.streamserve.com

Contents

Connector index

Input connectors

Output connectors

Command output connector

The Command output connector enables you to use commands to specify the output destination. The connector settings are described below.

Command

The command to execute. You can use a one-line command, or enter the path to a batch file or a script.

When passing output to an external application, use `%1` in the command string to specify the temporary path to the file containing the output data.

Example 1 *One-line command*

```
Physical "<[cmd=cat %1 > kv.$destination]>"
```

Example 2 *Path*

```
C:\project\myname.bat
```

Timeout

The time (seconds) to wait for a result. Timeout = 0 means wait forever.

14 Command output connector

Device input connector

This connector scans a UNIX FIFO (First-In-First-Out queue) for incoming jobs. The connector settings are described below.

Device path

The file path to the FIFO.

Example 3 *Device*

```
./fifos/My_Fifo
```

Schedule

Opens the Scheduler Configuration dialog where you specify when and how often to scan the FIFO. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.

Time-out

Specifies a time-out (seconds) for the connector. The StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, the StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

16 Device input connector

Directory input connector

This connector retrieves files from a named directory. The source application sends files to this directory, and the StreamServer retrieves the files.

Comments
Several connectors can scan the same directory for input. The connectors can belong to different StreamServer instances running on the same machine or on different machines. The Directory connector that first finds a match moves the file to a temporary directory. The reason is to prevent the file from being processed more than once. After the file is moved it is opened and processed. A temporary directory `streamserve_proc` is automatically created under the scanned directory. The default behavior is to move the files to this directory. If the StreamServer scans a directory located on a remote machine, it is safer to move the files to the `tmp` directory of the StreamServer (normally `export/tmp`). The reason is that if the network goes down, the StreamServer may also go down if the file is not stored locally.

The connector settings are described below.

Folder

The folder to scan for input. Use an absolute path, or a path relative to the working directory of the StreamServer. Two sub directories (`streamserve_lock` and `streamserve_proc`) are created when you run the Project. These directories ensure that a file is processed only once. These directories should not be tampered with.

File name pattern

The match criterion for the files to retrieve.

Example 4 *Pattern for retrieving all *.txt files*

```
*.txt
```

Interval for file size check

This interval specifies how frequently to check the file size. The file is not moved to the temporary directory immediately after it is found. First the file size is checked, and the file is moved only if the file size remains the same after this time interval. If the file size has changed, the connector waits for yet another interval to pass before it tries to move the file. This prevents the connector from moving the file before it is completed.

Sort by

StreamServer generates a list of all the files found in the scanned directory, and uses this list to find files that match **File name pattern** above. You can sort this list by:

- Name
- Date and time
- File extension
- File size

The files are by default sorted in descending order.

Sort ascending

Select to change the order in the sort list to ascending.

Save path

When a file is retrieved by the StreamServer it is removed from the scanned directory. If you want to save copies of the retrieved files you must specify a Save path. Use an absolute path, or a path relative to the working directory of the StreamServer.

Create Folder

If the input folder (path specified in **Folder** above) does not already exist, it is created by StreamServer if **Create Folder** is enabled.

Delete scanned files

If enabled, files are deleted from the scanned folder after they are retrieved by StreamServer. This ensures that an input file is not processed more than once.

If disabled, StreamServer will retrieve the files over and over again, using the polling interval specified (see **Schedule** below). This can be used, for example, to scan a database for changes once per polling interval.

Use spool directory as temporary storage

Select to use `streamserve_proc` as the temporary directory for the retrieved files. If the scanned directory is located on a remote machine, it is safer to use the `tmp` directory of the StreamServer (normally `export/tmp`). In this case you should not select this option.

Synchronize scanning

Select to enable synchronized scanning for the connector. The connector runs single-threaded and retrieves files one by one from the input folder.

If you do not select this option, the connector scans the input folder asynchronously. The connector runs multi-threaded and retrieves several files in parallel.

Schedule

Opens the Scheduler Configuration dialog where you specify when and how often to scan the directory. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.

20 Directory input connector

EmailIN input connector

This connector retrieves input sent via email. The connector settings are described below.

Mailbox type

The type of mailbox on the mail server.

Port

The port the StreamServer will use to communicate with the mail server.

Mail folder

The mail folder to scan for input. Leave this empty if you want to select the Inbox.

Mail server

The mail server IP address or host name.

User name

A user name for accessing the mailbox. If several StreamServer instances retrieve input from the same mail server, each instance must use a unique user name.

Password

A password for accessing the mailbox.

Read email body text

Select to enable processing of the email body. You can disable this option if you only want to process email attachments.

Retrieve email

- **Retrieve all** – Retrieve all emails in the mailbox.
- **Advanced** – Use the filter parameters below to specify which emails to retrieve. You can use wildcards.
  - **From** – Retrieve emails with specific From addresses.
  - **To** – Retrieve emails with specific To addresses.
  - **Cc** – Retrieve emails with specific Cc addresses.
  - **Date** – Use a timeframe to specify which emails to retrieve. The timeframe corresponds to the date and time the email was received. Use `YYYY-MM-DD` as format.
  - **Subject** – Retrieve emails with specific subjects.
  - **Reply to** – Retrieve emails with specific Reply to addresses.
  - **Request encryption** – Select to reject un-encrypted emails.

– **Request signature** – Select to reject unsigned emails.

Example 5 *Using the filter parameter Date*

Date: `2002-01-**`

Emails dated January 2002 will be retrieved.

Read attachment

Specifies which attachments to process.

Example 6 *Read attachment*

An email contains the attachments `INVOICE_101.xml`, `INVOICE_101.txt` and `COPY_101.xml`.

If **Read attachment file** is `*.xml`, the attachments `INVOICE_101.xml` and `COPY_101.xml` will be processed.

If **Read attachment file** is `INVOICE*.xml`, only `INVOICE_101.xml` will be processed.

Delete mail

- **No** – Do not delete emails.
- **Delete Processed** – Delete all retrieved and processed emails.
- **Delete all** – Delete all retrieved emails.
- **Advanced** – Use the filter parameters below to specify which retrieved emails to delete.
  - **Delete From** – Delete emails with specific From addresses.
  - **Delete To** – Delete emails with specific To addresses.
  - **Delete Cc** – Delete emails with specific Cc addresses.
  - **Delete Date** – Use a timeframe to specify which emails to delete. The timeframe corresponds to the date and time the email was received. Use `YYYY-MM-DD` as format.
  - **Delete Subject** – Delete emails with specific subjects.
  - **Delete Reply to** – Delete emails with specific Reply to addresses

Example 7 *Using the filter parameter Delete Date*

Delete Date: `2002-01-**`

Emails dated January 2002 will be deleted.

Save attachment

Select to save attachments to disk. The attachment files saved to disk will be given new unique names. The reason for this is to prevent files from being overwritten. You must use the following scripting functions where you map the original file names to the corresponding names of the files saved to disk:

- *GetAttachmentFile*
- *GetAttachmentOriginalFile*
- *GetAttachmentCount*

Example 8 *Mapping file names*

```
$i=1;
$count = GetAttachmentCount();
while(num($i)<=num($count))
{
  $original = GetAttachmentOriginalFile(num($i));
  if($original = "streamserve.gif")
  {
    $file = GetAttachmentFile(num($i));
  }
  $i++;
}
```

Attachments saved to disk are not removed automatically. One way to delete the attachments is to call an After Event script using the *FileDelete* scripting function.

Example 9 *Deleting attachments*

```
$i=1;
$count = GetAttachmentCount();
while(num($i)<=num($count))
{
  $delete = GetAttachmentFile(num($i));
  FileDelete($delete);
  $i++;
}
```

Attachment directory

The directory where to save the attachments.

Schedule

Opens the Scheduler Configuration dialog where you specify when and how often to try to retrieve emails. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.

Ignore email content

Select to ignore the email body and attachments. You can use this option to trigger an Event when an email is received, but not process any information in the email. In this case, you must use an all matching pattern in the Event (e.g. "?").

For example, use this option in "auto reply Projects", and use the *GetConnectorValue* script function to retrieve the appropriate email attributes (`From`, `Reply To`, etc.).

Additional scripting functions for attachment handling

Current attachment

You can use the *CurrInFileName* scripting function to fetch the file name of the current attachment sent through the StreamServer.

Content type and encoding of attachments

To retrieve content type and encoding of attachments you can use the following scripting functions.

- *GetAttachmentContentEncoding*
- *GetAttachmentContentType*

EmailIN attributes

You can use the scripting function *GetConnectorValue* to fetch EmailIN attributes.

Attribute	**GetConnectorValue("*<attribute>*")**
From	`GetConnectorValue("From")`
To	`GetConnectorValue("To")`
Cc	`GetConnectorValue("Cc")`
Reply To	`GetConnectorValue("Reply To")`
Subject	`GetConnectorValue("Subject")`
Date	`GetConnectorValue("Date")`
Body encoding	`GetConnectorValue("Encoding")`
Body type	`GetConnectorValue("Type")`
Attachment encoding	`GetConnectorValue("AttEncoding<n>")` where `<n>` is the attachment number.

Attribute	GetConnectorValue("<*attribute*>")
Attachment type	`GetConnectorValue("AttType<n>")` where `<n>` is the attachment number.
AttCount	`GetConnectorValue("AttCount")` Returns the number of attachments in the current email that have been saved to disk.

26 EmailIN input connector

Email output connectors

About sending emails

There are three types of configurations for sending emails:

- A standard Process and an email connector. See *Standard Process and email connector* on page 27.
- A MailOUT Process and an email connector. See *MailOUT Process and email connector* on page 28.
- A MailOUT Process, a PageOUT Process and a Topcall connector. See *MailOUT Process and Topcall connector* on page 30.

Standard Process and email connector

This type of configuration includes a standard Process (PageOUT Process, StreamOUT Process, etc.) and an email connector. Processed output will be sent directly from the Process, via the email connector, to a mail server. The email – address information, subject, etc. – is specified in the connector configuration. Which connector to use depends on the type of mail server. This is the most effortless method if you want to email processed output.

Email connectors

You can use the following email output connectors for this purpose:

- *MAPI output connector*
- *SMTP (MIME) output connector*

Attaching the processed output to the email

The processed output is sent as an email attachment. For example, if you select the PDF driver when you configure the email connector, the output will be attached as a PDF file. You specify attachment name, whether or not to compress the attachment, etc. in the Runtime settings for the email connector.

Specifying a mail server

You specify which type of mail server to send the output to when you select the email connector. For example, select an `SMTP (MIME)` connector if you want to use an SMTP mail server.

Email editor

The email – address information, subject, etc. – is specified in the Runtime connector configuration.

Attaching files on disk to the email

In the email editor you can also add supplementary attachments, such as product sheets, price lists, etc. These attachments must be available as files on disk.

Scripting functions for attachments

You can use the scripting functions *AttachmentBegin* and *AttachmentEnd* to handle attachments.

MailOUT Process and email connector

This type of configuration includes a MailOUT Process, an email connector and optionally a standard Process. The output from the MailOUT Process is an email. To this email you can attach files from disk and output from other Processes. The email is sent via an email connector to a mail server. Which connector to use depends on the type of mail server. This is the most effortless method if you only want to attach files on disk to an email, or send plain information in the email body.

Email connectors

You can use the following email output connectors for this purpose:

- *MAPI for MailOUT output connector*
- *SMTP (MIME) for MailOUT output connector*

Specifying a mail server

You specify which type of mail server to send the output to when you select the email connector. For example, select an `SMTP (MIME) for MailOUT` connector if you want to use an SMTP mail server.

Email editor

The MailOUT tool is the email editor. In the email editor you specify address information, subject, email body, etc. The same email editor is used for MAPI and SMTP MIME emails. This means that all parameters are not applicable to all email types. See *MailOUT Process reference* on page 29.

Attaching files on disk to the email

In the email editor you can add attachments to the email, for example product sheets, price lists, etc. These attachments must be available as files on disk.

Attaching processed output to the email

You can attach output from other Processes (PageOUT etc.) to the email. This Process must be connected to some kind of output connector. The driver options on that connector determines the format of the attachment. For example, if you select the PDF driver, the output will be attached as a PDF file.

Note: The Process that creates the attachment must be positioned above the MailOUT Process.

To attach processed output to the email

1. In the Runtime configuration view, right-click the MailOUT Process and select **Settings**. The Runtime Process Settings dialog box opens.
2. On the Attach Process tab, click **Add New**. The Process Attachment dialog box opens.
3. Enter the **Attachment name**.
4. From the **Select Process** drop-down list, select the appropriate Process and click **OK**.

To specify the content-type for the attachment

If you are using an SMTP mail server you can also specify the content-type for the attachment:

- **Autoselect** – the StreamServer determines which content-type to use. The information is retrieved from the driver that generates the attachment file.
- **Predefined** – select content-type from a predefine list.
- **Custom** – specify a custom content-type.

Scripting functions for attachments

You can use the scripting functions *AttachmentBegin* and *AttachmentEnd* to handle attachments.

MailOUT Process reference

From

Standard email attribute. Not applicable to MAPI.

Display name

The display name. Only applicable to SMTP (MIME). The From address is replaced by this name when the email is delivered to the recipient. This functionality must be supported in the email client.

Reply to

The address to reply to. Only applicable to SMTP (MIME). Used by the email recipient instead of the From address when responding to an email.

To

Standard email attribute.

Cc

Standard email attribute.

Bcc

Standard email attribute.

Subject

Standard email attribute.

Request receipt

Request a delivery receipt. A notification is received when the email is delivered. This functionality must be supported by the email servers and clients.

Attachments

Files to add as attachment to the email. Use the Add/Edit buttons to configure the settings. See *Mail attachment settings* below.

Mail attachment settings

Attachment

The path to the file to attach.

Attachment name

The name of the attachment. Leave this empty if you want the attachment name to be the same as the name of the source file.

Example 10 Attachment name

```
pricelist.pdf
```

MIME type

The MIME type (content-type) of the attachment. Only applicable to SMTP (MIME).

- **Predefined** – Select the MIME type from the **Value** drop-down list.
- **Custom** – Enter a custom MIME type in the **Value** field..

Convert attachment from PCL to PDF

Select to convert a PCL attachment to PDF.

Compress attachment

Select to compress the attachment.

MailOUT Process and Topcall connector

TOPCALL® is a registered trademark of TOPCALL International AG.

This type of configuration includes a MailOUT Process, a PageOUT Process, and a Topcall output connector. The PageOUT output is sent as a PDF or PCL attachment, via the MailOUT Process and the Topcall connector, to a designated directory. TOPCALL picks up the attachment and sends it via fax or email.

Specifying output directories

TOPCALL and the StreamServer use two directories to exchange information: one directory that contains the attachments, and one directory that contains address information. You specify these directories when you configure the Topcall connector.

MailOUT tool

In the MailOUT tool you must specify a **To** and **From** address. You can use dummy values.

Attaching processed output to the email

The PageOUT Process must be connected to some kind of output connector. The driver options on that connector determines the format of the attachment. For example, if you select the PDF driver, the output will be attached as a PDF file.

1. In the Runtime configuration view, right-click the MailOUT Process and select **Settings**. The Runtime Process Settings dialog box opens.
2. On the Attach Process tab, click **Add New**. The Process Attachment dialog box opens.
3. Enter the **Attachment name**.
4. From the **Select Process** drop-down list, select the appropriate Process and click **OK**.

MAPI output connector

This connector sends output via a MAPI mail server.

Comments
This connector is connected to a standard Process (PageOUT, StreamOUT, etc.). See *Standard Process and email connector* on page 27. In the Control Center, the logon options for the StreamServer service must not be set to `System account`. Use the logon option `This account` and specify an account with permissions to use the Windows Messaging Profile specified for the MAPI connector. To change the logon options, right-click the service in the Control Center and select **Service Startup**. **Note:** MAPI is not suitable for HTML messages. See http://support.microsoft.com/?id=268440

The connector settings are described below.

Default profile

The default Windows Messaging Profile.

A Windows Messaging Profile identifies a specific user and the mail server. To find the profiles available, open the Mail setup dialog box from the Control Panel and click **Show Profiles**.

The email address that will be used is the address specified as the From address in the user's profile. To be able to use alternative From addresses, you must set up a profile for each address. You do this in the runtime configuration of the connector.

Example 11 Default Profile

```
Outlook
```

Number of retries

The number of times to try to reconnect if the mail server does not respond.

Retry interval

The interval (seconds) between retries.

Runtime specific connector settings

Edit Mail

Opens the email editor. See *Edit mail settings* below.

Ignore missing attachments

Try to send an email with missing attachments.

Attachment name

The name of the attached Process output.

Example 12 Attachment name

```
$custno.pdf
```

Convert PCL to PDF

Select to convert PCL attachments to PDF.

Compress

Select to compress the attachments.

Code page

The code page for the email.

Profile name

The Windows Messaging Profile to use.

- **Use default Profile of connector** – Use the profile specified in the Platform configuration of the connector.
- **Static** – Use an alternative static profile.
- **Variable** – Use a variable to determine the alternative profile.
- **Lookup** – Use a lookup table to determine the alternative profile.

Edit mail settings

To, Cc, Bcc, and Subject

Standard email attributes.

Request receipt

Select to requesta delivery receipt. A notification is received when the email is delivered. This functionality must be supported by the email servers and clients.

Attachments

Supplementary files to add as attachment to the email. These attachments are files stored on disk, and not output from a Process. Use the Add/Edit buttons to configure the settings. See *Mail attachment settings* below.

Mail attachment settings

Attachment

The path to the file to attach.

Attachment name

The name of the attachment. Leave this empty if you want the attachment name to be the same as the name of the source file.

Example 13 Attachment

```
pricelist.pdf
```

Convert attachment from PCL to PDF

Select to convert a PCL attachment to PDF.

Compress attachment

Select to compress the attachment.

MAPI for MailOUT output connector

This connector sends output via a MAPI mail server.

Comments
This connector is connected to a MailOUT Process. See *MailOUT Process and email connector* on page 28. In the Control Center, the logon options for the StreamServer service must not be set to `System account`. Use the logon option `This account` and specify an account with permissions to use the Windows Messaging Profile specified for the MAPI for MailOUT connector. To change the logon options, right-click the service in the Control Center and select **Service Startup**. **Note:** MAPI is not suitable for HTML messages. See http://support.microsoft.com/?id=268440

The connector settings are described below.

Default profile

The default Windows Messaging Profile.

A Windows Messaging Profile identifies a specific user and the mail server. To find the profiles available, open the Mail setup dialog box from the Control Panel and click **Show Profiles**.

The email address that will be used is the address specified as the From address in the user's profile. To be able to use alternative From addresses, you must set up a profile for each address. You do this in the runtime configuration of the connector.

Example 14 *Default Profile*

```
Outlook
```

Number of retries

The number of times to try to reconnect if the mail server does not respond.

Retry interval

The interval (seconds) between retries.

Runtime specific connector settings

Profile type

The Windows Messaging Profile to use.

- **Use default Profile of connector** – Use the profile specified in the Platform configuration of the connector.

- **Static** – Use an alternative static profile.
- **Variable** – Use a variable to determine the alternative profile.
- **Lookup** – Use a lookup table to determine the alternative profile.

SMTP (MIME) output connector

This connector sends output via an SMTP mail server.

Comments
This connector is connected to a standard Process (PageOUT, StreamOUT, etc.). See *Standard Process and email connector* on page 27.

The connector settings are described below.

Mail server

The IP address or host name of the regular mail server.

Mail server user name

A user name for accessing the mail server.

Mail server password

A password for accessing the mail server.

Show password

When enabled (checked), **Mail server password** is displayed in plain text in the output connector dialog box. When disabled, each character in **Mail server password** is displayed as an asterisk (*). Note that this setting only affects how the password is displayed in the GUI.

Alternate mail server

The IP address or host name of an alternate mail server. Used if the regular mail server does not respond.

Domain name

The domain from which the StreamServer sends the email. The Domain Name must be a name accepted by the mail server. A mail server is normally configured to serve a specific domain.

Example 15 *Domain name*

```
streamserve.com
```

Number of retries

The number of times to try to reconnect if the mail server does not respond.

Retry interval

The interval (seconds) between the retries.

Sign

Select to sign emails (S/MIME). You must have created a security configuration for this purpose. See *Creating security configurations* in the *Encryption and authentication* documentation.

Encrypt

Encrypt emails (S/MIME). You must have created a security configuration for this purpose. See the *Encryption and authentication* documentation.

Runtime specific connector settings

Edit Mail

Opens the email editor. See *Edit mail settings* below.

Ignore missing attachments

Try to send an email with missing attachments.

Attachment name

The name of the attached Process output.

Example 16 Attachment name

```
$custno.pdf
```

Convert PCL to PDF

Select to convert PCL attachments to PDF.

Compress

Select to compress the attachments.

Attachment MIME Type

The MIME type (content-type) of the attached Process output.

- **Autoselect** – The StreamServer determines which MIME type to use. The information is retrieved from the driver that generates the attachment file.
- **Predefined** – Select the MIME type from the **Predefined MIME Type** drop-down list.
- **Custom** – Enter a custom MIME type in the **Custom MIME Type** field.

Code page

The code page for the email.

MIME Encoding

The MIME encoding for subject, body, and attachments.

- **Default** – Quoted Printable for subject and body. Base64 for attachments.

- **None** – No encoding for the subject. The maximum number of characters in the subject is 65. No encoding for the body. Base64 for attachments.
- **Quoted Printable** – Quoted Printable for subject, body, and attachments.
- **Base64** – Base64 for subject, body, and attachments.

S/MIME sign

Enable/disable signing of emails (S/MIME). Use static value or variable to enable (value=1) or disable (value=0).

You must have created a security configuration for this purpose. See *Creating security configurations* in the *Encryption and authentication* documentation.

S/MIME encrypt

Enable/disable encryption of emails (S/MIME). Use static value or variable to enable (value=1) or disable (value=0).

You must have created a security configuration for this purpose. See the *Encryption and authentication* documentation.

Edit mail settings

From, To, Cc, Bcc, and Subject

Standard email attributes.

Reply to

The address to reply to. Used by the email recipient instead of the From address when responding to an email.

Display name

The display name. The From address is replaced by this name when the email is delivered to the recipient. This functionality must be supported in the email client.

Request receipt

Select to request a delivery receipt. A notification is received when the email is delivered. This functionality must be supported by the email servers and clients.

Attachments

Supplementary files to add as attachment to the email. These attachments are files stored on disk, and not output from a Process. Use the Add/Edit buttons to configure the settings. See *Mail attachment settings* below.

Mail attachment settings

Attachment

The path to the file to attach.

Attachment name

The name of the attachment. Leave this empty if you want the attachment name to be the same as the name of the source file.

Example 17 Attachment name

```
pricelist.pdf
```

MIME Type

The MIME type (content-type) of the attachment.

- **Predefined** – Select the MIME type from the **Value** drop-down list.
- **Custom** – Enter a custom MIME type in the **Value** field..

Convert attachment from PCL to PDF

Select to convert a PCL attachment to PDF.

Compress attachment

Select to compress the attachment.

SMTP (MIME) for MailOUT output connector

This connector sends output via an SMTP mail server.

Comments
This connector is connected to a MailOUT Process. See *MailOUT Process and email connector* on page 28.

The connector settings are described below.

Mail server

The IP address or host name of the regular mail server.

Mail server user name

A user name for accessing the mail server.

Mail server password

A password for accessing the mail server.

Show password

When enabled (checked), **Mail server password** is displayed in plain text in the output connector dialog box. When disabled, each character in **Mail server password** is displayed as an asterisk (*). Note that this setting only affects how the password is displayed in the GUI.

Alternate mail server

The IP address or host name of an alternate mail server. Used if the regular mail server does not respond.

Domain name

The domain from which the StreamServer sends the email. The Domain Name must be a name accepted by the mail server. A mail server is normally configured to serve a specific domain.

Example 18 *Domain name*

```
streamserve.com
```

Number of retries

The number of times to try to reconnect if the mail server does not respond.

Retry interval

The interval (seconds) between the retries.

Sign

Select to sign emails (S/MIME). You must have created a security configuration for this purpose. See *Creating security configurations* in the *Encryption and authentication* documentation.

Encrypt

Select to encrypt emails (S/MIME). You must have created a security configuration for this purpose. See the *Encryption and authentication* documentation.

File output connector

This connector writes output to files on the file system. The connector settings are described below.

File

The file to write the output to. If the file does not exist, it will be created. You can specify the file using the *SetDestPath* scripting function.

Example 19 *File*

```
C:\projects\myfile.txt
```

Append

Select whether to append output to existing files.

- **Yes** – append the output to the file.
- **No** – overwrite the file.

Create directories

Select whether to create directories specified in the File path.

- **Yes** – If any of the directories in the File path does not exist, they are created.
- **No** – If any of the directories in the File path does not exist, an error message is generated.

File output connector

Fax connectors

Output via fax can be sent using the following connectors:

- *Fax Connect output connector*
- *TOPCALL output connector*

Third party products

StreamServe cannot take responsibility for the performance of third party products, or for any changes that might be made to those products by their respective manufacturers. For these reasons, this document does not provide specific information, illustrations or instructions for the use of third party products.

For specific information about any third party product, please refer to the manufacturer instructions for that product.

Fax Connect output connector

This connector sends output via different types of fax servers. You select the fax server when you specify the device driver. You can use the following fax servers:

- *Fax (Generic)*
- *RightFax and RightFax Unicode*
- *Zetafax*

The connector settings are described below

Type

- **Command** – Use a DOS / UNIX command, batch file, or script to specify the output destination.
- **Spool** – Send the output to a fax device.
- **File** – Send the output to a directory.

Command

The command specifying the destination for the output. Used with the type **Command**.

Example 20 Command

```
C:\project\myname.bat
```

Printer

The fax device. Used with the type **Spool**.

Directory

The output directory from which the fax device retrieves the output. Used with the type **File**.

Fax (Generic)

Fax (Generic) is a generic StreamServe fax driver.

Runtime specific connector settings

Fax Number

The fax number.

Priority

The priority of the output. The lower the number, the higher the priority. If you specify nothing, output will be delivered on a first-in, first-out basis.

Comment 1-3

Comments to add to the cover page.

Cover Page

The path to the cover page. Maximum 8 characters.

Alt. Destination

The fax number to an alternate destination. Used if the StreamServer cannot connect using the standard fax number.

User

The name of the sender.

Send Time

The time when to send the fax. Use the format `HH[:MM:SS]`. If you specify nothing, the fax will be sent immediately.

RightFax and RightFax Unicode

RightFAX® is a registered trademark of the AVT Document Exchange Software Group.

RightFax Unicode

RightFAX Unicode enables printing of all TrueType fonts that have been specified in the `RightFaxUC.drs` file. The fonts do not have to be installed on the fax system. Unicode is only supported for the fax body, not for cover pages etc.

Runtime specific connector settings

See your RightFax documentation.

Zetafax

Zetafax® is a registered trademark of Equisys.

Runtime specific connector settings

See your Zetafax documentation.

FTP connectors

The *FTP input connector* and *FTP output connector* enables the StreamServer to transfer files to and from an FTP server.

FTP input connector

This connector retrieves files from an FTP server.

Comments
When a file has been retrieved, it is deleted from the FTP server. If a Project has several connectors (FTP input or FTP output) per user account, the number of concurrent connections specified on the FTP server must be equal to, or more than, the number of FTP connectors. For example, if a Project has one FTP input connector and one FTP output connector that use the same user account, the number of concurrent connections specified on the FTP server must be at least 2.

The connector settings are described below.

Server

The FTP server to connect to.

Example 21 *Server*

```
rtu.abx.com
```

Port

The port on the FTP server.

User name

A user name for accessing the FTP server.

Password

A password for accessing the FTP server.

Remote directory

The directory to be accessed on the FTP server. If you leave this blank, the root directory will be set as remote directory. The FTP server software determines whether to use slash or back-slash.

Example 22 Remote directory

```
/invoices/pdfout/area2
```

File mask

The match criterion for the files to be retrieved.

Example 23 File mask

```
*.pdf
```

Data transfer mode

Specifies the transfer mode for the data.

- **Active** – use active mode.
- **Passive** – use passive mode. This mode is used when communicating through firewalls. It opens a control connection to the FTP server, tells the FTP server to expect a control connection and a second connection. Then it opens the data connection to the FTP server on a randomly chosen high-numbered port. This works with most firewalls unless the firewall restricts outgoing connections on high-numbered ports.

File transfer type

Specifies the transfer mode for the files.

- **ASCII** – use ASCII, Type A, transfer method. Control and formatting data is converted to local equivalents.
- **Binary** – use FTP Image, Type I, transfer method. The file is transferred without any changes.

Schedule

Opens the Scheduler Configuration dialog where you specify when and how often to try to retrieve files. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.

Time-out

Specifies a time-out for the connector. The StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, the StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

FTP output connector

This connector transfers files to an FTP server.

Comments
If a Project has several connectors (FTP input or FTP output) per user account, the number of concurrent connections specified on the FTP server must be equal to, or more than, the number of FTP connectors. For example, if a Project has one FTP input connector and one FTP output connector that use the same user account, the number of concurrent connections specified on the FTP server must be at least 2.

The connector settings are described below.

Server

The FTP server to connect to.

Example 24 *Host*

```
rtu.abx.com
```

Port

The port on the FTP server.

User name

A user name for accessing the FTP server.

Password

A password for accessing the FTP server.

Remote directory

The directory to be accessed on the FTP server. If you leave this blank, the root directory will be set as remote directory. The FTP server software determines whether to use slash or back-slash.

Example 25 *Remote path*

```
/invoices/pdfout/area2
```

Create Directories

Select whether to create the Remote directory if it does not exist.

- **Yes** – If any of the directories in the remote path does not exist, they are created.

- **No** – If any of the directories in the remote path does not exist, the file is put in the undeliverables folder.

File

The file to write the output to, and to upload to the FTP server. For example:

```
$custno.pdf
```

Tmp File

A temporary filename for the file to upload.

You must use a file name pattern different from the filename pattern used by the receiving application. This prevents the receiving application from downloading the file before the upload is completed. When the upload is completed, the file is renamed to the name specified in **File** above, and the receiving application can download the file.

If external files are uploaded (see **External files** below) all files will be uploaded, one by one, with the same temporary file name. After a successful upload, each file will be renamed to its final file name.

External Files

Enables upload of external files, together with the generated output file, to the **Remote directory** specified.

You can use an absolute path to the external file, or a path relative to the working directory of the StreamServer application.

You can add several external files if you separate the file paths with semicolon. For example:

```
<path1>;<path2>;<path3>
```

Data transfer mode

Specifies the transfer mode for the data.

- **Passive** – use passive mode. This mode is used when communicating through firewalls. It opens a control connection to the FTP server, tells the FTP server to expect a control connection and a second connection. Then it opens the data connection to the FTP server on a randomly chosen high-numbered port. This works with most firewalls unless the firewall restricts outgoing connections on high-numbered ports.
- **Active** – use active mode.

File transfer type

Specifies the transfer mode for the files.

- **ASCII** – use ASCII, Type A, transfer method. Control and formatting data is converted to local equivalents.
- **Binary** – use FTP Image, Type I, transfer method. The file is transferred without any changes.

Runtime specific connector settings

The `<Default>` options for Passive Mode and Transfer Mode means the Platform settings will be used.

Generic Archive output connector

This connector archives output (output data and corresponding index files) in an external archiving system. The output is temporarily stored in directories before it is sent to the archiving system. Each index file contains the path to the output data in relation to the directory where the index file is stored.

Comments
What settings to specify depends on the requirements from the archiving system. For example, if an archiving client scans the directory for index files with a certain extension, you must specify this extension. As the extension is added at job end, you prevent the archiving client from accessing an index file before it is completed.

The connector settings are described below.

Index directory

The root directory for the output. The directory is specified in relation to the working directory. You can use the `SetDestPath()` scripting function.

One sub directory is automatically created per job, containing the output data and the index file. As an alternative to these automatically generated directories, you can specify separate directories for the output data and the index files in the Runtime specific connector settings.

Runtime specific connector settings

Index entry

The characters (hexadecimal values) used to separate the **Archive attributes** within an index line.

Index line separator

The characters (hexadecimal values) used to separate index lines.

Index command

A command to execute when all files are added to the **Index directory** in the Platform connector settings.

Example 26 Index command

```
programname -st %l -dt %
```

where *programname* retrieves the index file (`%1`) in the **Index directory** (`%`) for further processing.

Remove job directory

Removes the job directory (in the Index directory in the Platform connector settings) after successful completion of the **Index command**.

Index directory

A separate directory for the index files. The directory is specified in relation to the Index directory in the Platform connector settings. You can use variables. If no directory is specified, the job sub directory in the Index directory in the Platform connector settings is used.

Index name

The name of the index file. You can use variables. If no name is specified, a name is automatically generated.

Index extension

An extension, to be added to the index file. You can use variables. If no extension is specified, the `*.idx` extension is used.

Data directory

A separate directory for the output data. The directory is specified in relation to the Index directory in the Platform connector settings. You can use variables. If no directory is specified, the output data is stored in the same directory as the index file.

Data name

The name of the output data. You can use variables. If no name is specified, a name is automatically generated.

Archive attributes

The archive attributes. These attributes consist of an attribute name and a value. The value can be a static text or a variable. By default, there is an attribute for the Process output file name.

You must manually add any other attributes. The order in which the attributes are listed is important since StreamServer outputs the attributes in this order.

You can enter the attribute name `<idxcodepage>` and the appropriate code page as attribute value to specify a code page for the index file. The attribute name must include the < and > characters.

HTTP connectors

The *HTTP(S) input connector* enables the StreamServer to function as an HTTP server.

The *HTTP(S) Poll input connector* enables the StreamServer to function as an HTTP client that polls an HTTP server.

The *HTTP(S) Submit output connector* enables the StreamServer to function as an HTTP client that submits output to an HTTP server.

The *HTTP Response output connector* enables the StreamServer to respond to an HTTP request.

HTTP(S) input connector

These connectors enable the StreamServer to function as an HTTP server.

Comments
• Use the HTTPS connector if you want the StreamServer to function as an SSL server that communicates over an encrypted HTTPS channel. • An HTTP(S) connector must be connected to an input queue. • You can use the custom startup argument `-startallinconnectors` to start connectors that are not connected to any Event.

The connector settings are described below.

Security configuration

The security configuration to add to the HTTPS connector. See *Creating security configurations* in the *Encryption and authentication* documentation for more information about security configurations. The security configuration must be included in a resource set connected to the Platform.

Version

The HTTP version to use. **Auto** means the version is determined by the client.

SSL version

The SSL version to use with the HTTPS connector. The server and the clients must use the same SSL version.

Expose in Collector

Select to expose the connector as a reprocess service in the Collector web application.

Description

The description of the connector shown in the Collector web application.

Example 27 Description

```
Fax invoices
```

Reprocess type

The output channel for the reprocess service.

Address

An alternative network address for the StreamServer, for example the IP address to a specific network card. You can leave this empty if you want to use the default network address for the workstation.

Port

The port the connector listens to for HTTP requests. If the Project contains several HTTP(S) connectors, you must select a unique port for each connector. Instead of using several HTTP(S) connectors, you can use one HTTP(S) input connector and different URIs to other types of input connectors. See *HTTP Realms tab* on page 60.

Input threads

The maximum number of concurrent connections. When all connections are busy, and a new client tries to connect to the StreamServer, the connection may fail. Increasing the number can decrease performance.

Idle time-out

A time-out (milliseconds) applied when the StreamServer has finished processing a request, and no more data related to the request will be received or sent. This time-out sets the maximum time the connection will remain open, and enables the client to send a new request without having to set up a new connection.

Time-out

A time-out (milliseconds) applied when the StreamServer sends or receives data. If no data is sent or received during the time specified (dead connection), the connection will be closed.

Response time-out

Response time-out is disabled by default, and the client is expected to wait until StreamServer has finished processing the data before it receives a response. To enable Response time-out, you must add the custom keyword `UseResponseTimeouts` to `Platforms` > *`platform name`* > `Logical`. See *Using custom commands and keywords* in the *Design Center* documentation for more information about how to add custom commands and keywords.

Response time-out is a time-out (milliseconds) applied when the StreamServer has received all data from the client. This time-out sets the maximum time the client is expected to wait for a response.

The client can use an HTTP header field to override this time-out. To enable this, you must use the keyword `HTTPResponseTimeOut`. With this keyword you can also specify a file to return to the client when the time-out occurs. See *Custom HTTP(S) connector settings* on page 62.

If the client does not expect any response from the StreamServer, you should set this time-out to "0". In this case the StreamServer will immediately return "200 OK", and optionally a file using `HTTPResponseTimeOut`, to inform the client that the request has succeeded.

Authentication

The type of authentication scheme (RFC 2617) to use for password authentication. The authentication scheme you specify here applies to all HTTP realms you specify for the HTTP(S) connector. See *HTTP Realms tab* on page 60.

- **None** - Do not use authentication.
- **Basic** - Send authentication parameters as clear text. This is the only scheme supported in HTTP/1.0.
- **Digest** - Send authentication parameters as a checksum over the network. Requires HTTP/1.1.

Publish directory

The root directory for stored files. If you want to enable clients to access stored files via HTTP(S), you must specify a publish directory.

Publish extension file

A file that associates file formats and content-types (RFC 2045). Applies to files in the Publish directory. The StreamServer accepts `html`, `htm`, `gif`, `jpg`, `txt`, `zip` by default. To use other formats, you must specify a publish extension file.

File syntax:

`Target ContentType CustomHeader`

Where *`Target`* is the format to associate with a content type, *`ContentType`* is the content type, and *`CustomHeader`* is an optional custom header (*`Name`*`:`*`Value`*).

Example 28 *Publish extension file*

```
.pdf application/pdf
/qwerty.tbl text/plain
```

Line 1 associates all `*.pdf` files with content type `application/pdf`.

Line 2 associates the file `qwerty.tbl` with content type `text/plain`.

Job resource URI

Identifies output files stored via a Job Resource output connector. Used if the client expects a response to be presented in a web browser via an ActiveX plug-in. In such a case, the response must be temporarily stored using a Job Resource output connector. See *Responding with processed data via a Job Resource connector* on page 72.

Example 29 Job resource URI

```
/jr
```

HTTP Access tab

On this tab you specify the HTTP connector and URI to access the connector. See the *Design Center* documentation for more information about HTTP Access.

HTTP connector

Select this connector from the drop-down list.

URI

Specify the URI to access the connector.

HTTP Realms tab

On the HTTP Realms tab you can specify several realms for the HTTP(S) connector. A realm is defined by one or more URIs. To access a realm, clients must provide a valid user name and password. All these user names and passwords are specified in a password file. Click **Add** or **Edit** to specify the settings for each realm.

Name

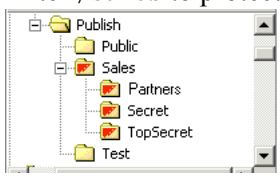
The name of the realm.

URI(S)

The URI or URIs for the realm. Multiple URIs are separated by comma.

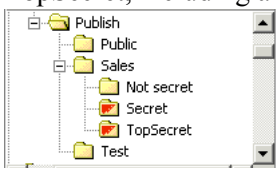
Example 30 Password protecting parts of the publish directory (single URI)

Enter `/Sales` to protect the folder Sales and all its sub-folders.



Example 31 *Password protecting parts of the publish directory (multiple URIs)*

Enter `/Sales/Secret,/Sales/TopSecret` to protect the folders Secret and TopSecret, including all sub-folders.

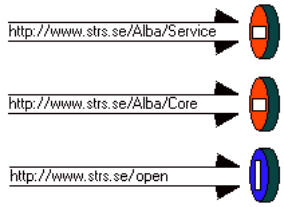


Example 32 *Password protecting connectors - 1*

Enter `/Alba` to protect a connector with HTTP access via the URI `/Alba`.

Example 33 *Password protecting connectors - 2*

Enter `/Alba` to protect two connectors with HTTP access via the URIs `/Alba/Service` and `/Alba/Core`.



Password file

The password file resource that contains the user names and corresponding passwords. The password file has the following syntax:

user_name [tab] *password*

Use ASCII and do not include blank spaces.

Custom HTTP(S) connector settings

You can add custom keywords to the HTTP and HTTPS connectors. See *Using custom commands and keywords* in the *Design Center* documentation for more information about how to add custom commands and keywords.

`HTTPExtJobIdField "ID_name";`
Retrieves the external ID from an application that sends input to the StreamServer. The application that sends the input must include the external ID as HTTP header information (`ID_name:ID_value`).

`HTTPMaxContentLength limit;`
Limits the size (in bytes) of the body in an HTTP request. If a request contains a body that exceeds this limit, an error response will be returned.

`HTTPResponseTimeOut "timeout" "path" "custom" END;`

Specifies what the StreamServer should do when a response time-out occurs.
timeout:
A time-out set by the client to override the response time-out. For example, the client header field "`x-timeout:60000`" sets the response time-out to one minute. To enable the client to set the response time-out, you can enter
`HTTPResponseTimeOut "x-timeout" "" "" END;`
Leave the `timeout` argument empty if you do not want to allow clients to set the response time-out.
path:
A file with information to send in the response when the time-out occurs. For example:
`HTTPResponseTimeOut "" "/Messages/timeout.txt" "" END;`
custom:
Custom fields (`name:value`) added to the response header. For example:
`HTTPResponseTimeOut "" "timeout.txt" "content-type:text/plain" END;`
You can separate several `name:value` pairs with `<0d><0a>`.

HTTP(S) Poll input connector

These connectors enable the StreamServer to function as an HTTP client that polls an HTTP server.

Comments
• Use the HTTPS Poll connector if you want the StreamServer to function as an SSL client that communicates over an encrypted HTTPS channel. • You can use the custom startup argument `-startallinconnectors` to start connectors that are not connected to any Event.

The connector settings are described below.

Security configuration

The security configuration to add to the HTTPS Poll connector. See *Creating security configurations* in the *Encryption and authentication* documentation for more information about security configurations. The security configuration must be included in a resource set connected to the Platform.

HTTP method

The method to use when polling the HTTP server.

- **HEAD** – Requests a HEAD header from the HTTP server. If the `Last-Modified` entity-header field has changed, the StreamServer will fetch and process the data. If there is no `Last-Modified` field in the header, the connector will switch to the GET method instead. If the `Last-Modified` field is not updated correctly this method will fail.
- **GET** – Downloads the data once during each poll interval, and calculates a checksum to see if the data has been updated. The StreamServer will process the data only if the checksum has changed.
- **POST** – Posts a file to the HTTP server, downloads the response and calculates a checksum. The StreamServer will process the data only if the checksum has changed.

URL

The URL to the HTTP server.

Content-type

This property depends on the HTTP method:

- **POST** – The content type of the posted file.
- **HEAD** and **GET** – The media types the client accepts in the response. For example `text/*, text/xml, text/xml;level=1, */*`. All formats will be accepted if this field is empty.

Post file name

The file to post when using HTTP method POST. The file must be included in a resource set connected to the Platform.

HTTP time-out

The maximum time (milliseconds) to wait before aborting a transfer.

HTTP version

The HTTP version to use.

SSL version

The SSL version to use with the HTTPS Poll connector. The server and the clients must use the same SSL version.

HTTP authentication

The type of authentication scheme (RFC 2617) to use for password authentication.

- **None** - Do not use authentication.
- **Basic** - Send authentication parameters as clear text. This is the only scheme supported in HTTP/1.0.
- **Digest** - Send authentication parameters as a checksum over the network. Requires HTTP/1.1.

HTTP realm

The name of the realm to access. Used only if authentication is required.

User name

A user name to access the realm. Used only if authentication is required.

Password

A password to access the realm. Used only if authentication is required.

Cache file

The cache file that stores communication checksums between StreamServer start/stop.

Schedule

Opens the Scheduler Configuration dialog where you specify when and how often to poll the HTTP server. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.

Time-out

Specifies a time-out for the connector. The StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, the StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

HTTP(S) Submit output connector

These connectors enable the StreamServer to function as an HTTP client submitting output to an HTTP server.

Comments
Use the HTTPS Submit connector if you want the StreamServer to function as an SSL client that communicates over an encrypted HTTPS channel.

The connector settings are described below.

Use security configuration

Select whether to use a security configuration or a CA certificate for the HTTPS Submit connector. If the HTTPS server requires client authentication, you must use a security configuration. If the HTTPS server does not require client authentication, you can use a CA certificate instead.

CA certificate

The CA root certificate that confirms the identity of the SSL server. See *Creating security configurations* in the *Encryption and authentication* documentation for more information about security configurations. The certificate must be included in a resource set connected to the Platform.

PEM encoded x 509 certificates are supported.

Security configuration

The Security configuration to use with the HTTPS Submit connector. See the *Encryption and authentication documentation* for more information. The security configuration must be included in a resource set connected to the Platform.

Method

The method to use when submitting output to the HTTP server.

- **POST** – Send output to the HTTP server for further processing.
- **PUT** – Send output to the HTTP server. For example, if output is a web page, select PUT to put it on a web server. Requires scripting and specific access rights to the server.

URL

The URL to the HTTP server.

Content-type

The content-type of the output to send to the HTTP server.

Example 34 Content-type

```
text/html; charset="ascii"
```

Time-out

The maximum time (milliseconds) the StreamServer waits before cancelling a transfer.

Version

The HTTP version to use.

SSL version

The SSL version to use with the HTTPS Submit connector. The server and the clients must use the same SSL version.

Authentication

The type of authentication scheme (RFC 2617) to use for password authentication.

- **None** - Do not use authentication.
- **Basic** - Send authentication parameters as clear text. This is the only scheme supported in HTTP/1.0.
- **Digest** - Send authentication parameters as a checksum over the network. Requires HTTP/1.1.

Realm

The name of the realm to access. Used only if authentication is required.

User name

A user name to access the realm. Used only if authentication is required.

Password

A password to access the realm. Used only if authentication is required.

Cache file

The cache file that stores communication checksums between StreamServer start/stop.

Runtime specific connector settings

Custom header

A custom header to add to the output. Use the following syntax:

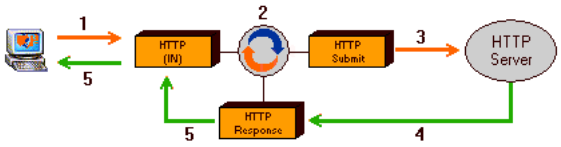
Name:*Value*

You can separate *Name*:*Value* pairs with <0d><0a>.

Response place connector

The name of the connector that receives the response from the HTTP server. Used only if a response is required. This connector is either an HTTP Response output connector, or any type of queue enabled input connector.

Example 35 Using an HTTP Response connector as Response place connector



1 A client sends input to the StreamServer via an HTTP input connector.

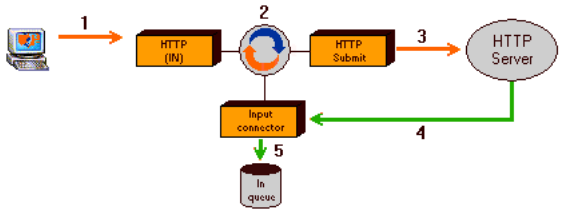
2 The StreamServer processes the data.

3 The processed data is sent via an HTTP Submit connector to an HTTP server.

4 The HTTP Response connector collects the response from the HTTP server.

5 The HTTP connector retrieves the response from the HTTP Response connector, and sends it to the client.

Example 36 Using an input connector as Response place connector



1 A client sends input to the StreamServer via an HTTP input connector.

2 The StreamServer processes the data.

3 The processed data is sent via an HTTP Submit connector to an HTTP server.

4 The input connector collects the response from the HTTP server.

5 The response data is queued as a separate job.

Response custom header

A custom header to add to the response from the HTTP server. Use the following syntax:

```
Name:Value
```

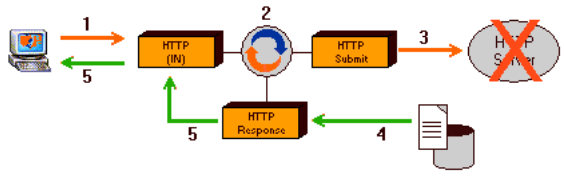
On failure file

The path to a file to be used if output cannot be delivered to the HTTP server.

On failure place connector

The connector that retrieves the On failure file specified above. This connector is either an HTTP Response output connector, or any type of queue enabled input connector. For example, if the StreamServer receives input from a client, and sends output to an HTTP server, an error message (the on failure file) can be sent back to the client via an HTTP Response connector.

Example 37 Using an HTTP Response connector as On failure place connector



1 A client sends input to the StreamServer via an HTTP input connector.

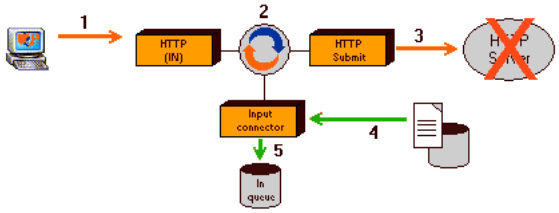
2 The StreamServer processes the data.

3 The StreamServer tries, but fails, to send the processed data via an HTTP Submit connector to an HTTP server.

4 The HTTP Response connector collects the On failure file.

5 The HTTP connector retrieves the On failure file from the HTTP Response connector, and sends it to the client.

Example 38 Using an input connector as On failure place connector



1 A client sends input to the StreamServer via an HTTP input connector.

2 The StreamServer processes the data.

3 The StreamServer tries, but fails, to send the processed data via an HTTP Submit connector to an HTTP server.

4 The input connector collects the On failure file.

5 The On failure file is queued as a separate job.

On failure content-type

The content type of the On failure file.

Example 39 On failure content type

```
text/html; charset="ascii"
```

On failure custom header

A custom header to add to the response with the On failure file. Use the following syntax:

Name:Value

HTTP Response output connector

This connector enables the StreamServer to respond to an HTTP request. The connector settings are described below.

Content-type

The content-type of the response.

Example 40 Content-type

```
text/html; charset="ascii"
```

Runtime specific connector settings

Custom header

A custom header to add to the output. Use the following syntax:

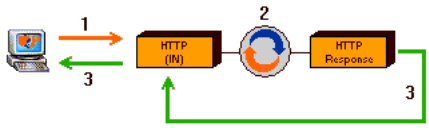
Name:*Value*

You can separate *Name*:*Value* pairs with <0d><0a>.

HTTP connector scenarios

Responding with processed data via an HTTP Response connector

The HTTP(S) connector enables the StreamServer to act as an HTTP server that receives and processes input from an HTTP client. You can configure the StreamServer to automatically return processed output to the client. In most cases you will use an HTTP Response output connector to handle the response.



1. A client sends input to the StreamServer via an HTTP input connector.
2. The StreamServer processes the data.
3. The processed data is sent back to the client via the HTTP Response and HTTP connectors.

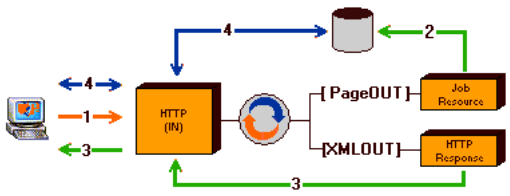
Configuring the HTTP Response connector

On the HTTP Response connector, you must specify which driver to use, and corresponding content-type (RFC 2045) for the response. You can do this in the Platform or in the Runtime configuration.

Responding with processed data via a Job Resource connector

The HTTP(S) connector enables the StreamServer to act as an HTTP server that receives and processes input from an HTTP client. You can configure the StreamServer to automatically return processed output to the client.

If the client expects the response, for example a PDF file, to be presented in a web browser via an ActiveX plug-in, the response must be temporarily stored at the server side.



1. A client sends input to the StreamServer via an HTTP input connector.
2. The StreamServer processes the data and sends PDF output via a Job Resource connector to a temporary storage.

3 An XMLOUT Process generates HTML output with information specifying the location of the processed data. This output is sent back to the client via the HTTP Response and HTTP connectors.

4 The client sends a GET request via the HTTP input connector, and receives the processed data with the response.

Specifying the URI to the temporary storage

1 Open the Input Connector Settings dialog box for the HTTP connector.

2 In the **Job resource URI** field, enter `/JobResource`.

Fetching the URI for the temporarily stored output

Add the following After Process script to the PageOUT Process:

```
$GetJr = GetJobResourceURI();
```

Note: Add the script only to the first PageOUT Process if you include output from multiple PageOUT Processes.

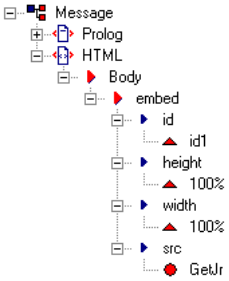
Configuring the XMLOUT Process

The XMLOUT Process generates HTML output that uses the `<embed>` tag to specify the location of the processed data:

```
<embed id="ID" height="X%" width="Y%" src="Variable"/>
```

Example 41 XMLOUT Process

Elements and attributes:



Output

```
<?xmlversion="1.0">
<HTML>
   <Body>
      <embed id="id1" height="100%" width="100%" src="$GetJr"/>
   </Body>
</HTML>
```

Configuring the Job Resource connector

1 Open the Output Connector Settings dialog box for the Job Resource connector.

2 Click the **Driver** icon and select **Device** > **PDF**.

3 Click the Connector icon and specify the storage settings.

Storage settings	
Name	`invoice.pdf`
Content-type	`application/pdf`
Expire by time	`Yes`
Duration	`30`
Time unit	`Minute`
Expire by access	`No`
Expire with Job	`No`

Configuring the HTTP Response output connector

1 Open the Output Connector Settings dialog box for the HTTP Response connector.

2 In the **Content-type** field, enter `text/html`.

Mailing a link to the output

An alternative to the XMLOUT Process and HTTP Response output connector is to use a MailOUT Process and email connector to send a link to the output.

The recipient can click the link to open the document. Use the following syntax for the link in the email body:

`http://<host>:<port><variable>`

where `<variable>` is the variable specified in the script, for example `$GetJr`.

Example 42 Specifying the link when the mail format is Plain Text.

```
http://wxbgaje02:81$GetJr
```

Example 43 Specifying the link when the mail format is HTML.

```
<a href="http://wxbgaje02:81$GetJr">Open document</a>
```

Note: The Job Resource connector settings `Expire by time` and `Expire by access` determines how long the document will live.

Internal connectors

The *Internal input connector* enables the StreamServer to transfer data internally.

The *Internal output connector* enables the StreamServer to loop back output to an Internal input connector for further processing.

Internal input connector

This connector enables the StreamServer to transfer data internally. For example to loop back output via an Internal output connector for further processing, or to pick up input received via an HTTP input connector.

Comments
If you loop back output via an Internal output connector, you must use the UTF-8 code page for both connectors.

There are no connector settings for this type of connector.

Internal output connector

This connector enables the StreamServer to loop back output to an Internal input connector for further processing.

Comments
Data sent from the Internal output connector must not contain any formatting information. You must use the UTF-8 code page for both the Internal input and output connectors. An Internal output connector always runs in output mode Job.

The connector settings are described below.

Destination (input connector name)

The Internal input connector to send the output to.

JDBC connectors

The JDBC input and output connectors provide a read-write interface for relational databases. The *JDBC input connector* is used to retrieve information from the database, and the *JDBC output connector* is used to insert or update information in the database.

In the configuration of the JDBC input and output connector you specify the appropriate JDBC driver and Connection URL settings to connect to the database, and the SQL query to execute on the selected database.

Prerequisites

- The following environment variable must be set:

  ```
  STRS_JAVA_HOME="C:\Program Files\Java\jre1.5.0_14"
  ```

- The appropriate JDBC driver must be available to connect to the database.

JDBC drivers included in the StreamServe installation

The following JDBC drivers are included in the StreamServe installation:

- JDBC-ODBC bridge
- DataDirect SQL Server driver
- DataDirect Oracle driver

Downloading JDBC drivers

You can also download java packages with other drivers, and set the classpath to the downloaded `*.jar` file.

You use the argument *-java-user-class-path* in the Configure Platform Export dialog box (Administrator Arguments) to set the classpath.

JDBC input connector

The JDBC input connector retrieves data from relational databases. In the connector configuration you specify the appropriate JDBC driver and Connection URL settings to connect to the database, and the SQL query to execute on the selected database.

The SQL query is executed at a time interval (Polling interval) specified in the connector settings. This means the JDBC input connector uses the same time interval when retrieving data from the database.

Comments
• This connector must be used together with a MessageIN Event. • The MessageIN Event must use an SXD file with one field element for each column in the database table stated in the SQL query.

The connector settings are described below.

Connector type

Select `Java`.

Java class

Select `JDBCInConnector`. When you click the button attached to this field, all available Java input connector classes are displayed. To activate the JDBC input connector settings you must select the java class `JDBCInConnector`.

JDBC driver

The JDBC driver to use to connect to the database. You can select the following drivers from the drop-down list:

To use this driver...	Select this option
JDBC-ODBC bridge	`sun.jdbc.odbc.JdbcOdbcDriver`
DataDirect SQL Server driver	`com.streamserve.www.jdbc.sqlserver.SQLServerDriver`
DataDirect Oracle driver	`com.streamserve.www.jdbc.oracle.OracleDriver`

If you are using other drivers than the above, you can enter the corresponding driver expression in this field.

Connection URL

The connection URL to connect to the database. You can select and edit the following Connection URLs from the drop-down list:

If you are using this driver...	Select and edit this option

JDBC-ODBC bridge	**Option:**
	`jdbc:odbc:DB`
	Example:
	`jdbc:odbc:myDatabase`

DataDirect SQL Server driver	**Option:**
	`jdbc:streamserve:sqlserver://server[:port];databaseName=DB`
	Example:
	`jdbc:streamserve:sqlserver://localhost:1433;databaseName=myDatabase`

DataDirect Oracle driver	**Option:**
	`jdbc:streamserve:oracle://server[:port];SID=DB`
	Example:
	`jdbc:streamserve:oracle://localhost:1433;SID=myDatabase`

If you are using other drivers than the above, you can enter the corresponding Connection URL in this field.

User name

A user name to access the database. Overrides any other user names specified in the Connection URL.

Password

A password to access the database. Overrides any other passwords specified in the Connection URL.

SQL file

You can create an SQL file where you enter the query to submit to the database. This query overrides any query specified in the **SQL command** field described below.

If you use the **SQL file** option to specify the query, you can create more complex queries using several lines of statements.

Example 44 Query in SQL file

```
SELECT *
FROM CustomerData
```

```
WHERE CustNo>200
```

SQL command

You can enter a one-line query in the SQL command field.

Example 45 Query in SQL command

```
SELECT * FROM CustomerData WHERE CustNo>200
```

Event name

The name of the MessageIN Event that retrieves the data via this input connector.

Mode

Specifies what to do with each table row in the database table after it is read.

Move (with delete)	Delete the table row after it is read.
Copy (no delete)	Do not delete the table row after it is read.

Row level

Specifies how to handle each table row retrieved from the database table when delivering data to the MessageIN Event.

Objects as blocks	Each row is added as a block instance to the MessageIN Event. Using this option, the input job starts when the first row is retrieved and ends when the last row is retrieved. All rows are added as block instances to the same MessageIN Event.
Objects as events	Each row triggers a new MessageIN Event. Using this option, the input job also starts when the first row is retrieved and ends when the last row is retrieved. Each row triggers a new MessageIN Event.
Objects as jobs	Each row starts a new input job. Using this option, each row represents a separate input job and triggers a new MessageIN Event.

Create SXD

Used when creating an SXD file. Enter the path to the SXD file to create, for example:

```
C:\resources\SXD\JDBCretrieve.sxd
```

See *Creating an SXD file* on page 82 for more information.

License Password

Password for the JDBC driver (if required). If you are using any of the DataDirect JDBC drivers included in the StreamServe installation you should leave this field blank.

Time-out

Specifies a time-out for the connector. StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

Polling interval

The interval at which the SQL query is submitted to the database, and at which this connector retrieves data from the same database.

Creating an SXD file

The MessageIN Event that retrieves data via the JDBC input connector must use the appropriate SXD file to be able to handle the data. This SXD file must include one field element for each column in the database table from which data is retrieved.

You can let the JDBC input connector retrieve the table information, and create this SXD file for you. You can create a separate Project for this purpose. This Project must contain a JDBC input connector, a MessageIN Event, and any Process and output connector. For example a PageOUT Process containing a dummy text field and a Null output connector.

When you run the Project, the SXD file is created in the path you specify in the connector settings. You must then import the SXD file to the appropriate resource set, and connect it to the MessageIN Event in the real Project.

JDBC input connector configuration

When you configure the connector to create an SXD file, you must specify the connection settings (JDBC driver, Connection URL, User name, and Password) to connect to the database with the table from which to retrieve the information. You can use the same connection settings as in the real Project, but if you need to you can connect to another database that contains a table with identical columns as the one you will use in the real Project.

To create the SXD file, you must also specify the following:

SQL file or SQL command	The SQL query must select all columns in the database table. For example, to select all columns in the table `CustomerData` you can use the following SQL command: `SELECT * FROM CustomerData`
Event name	The same Event name as in the real Project.
Mode	Select **Copy (no delete)**. This will make sure no data is removed from the database table.
Row level	The same row level as in the real Project. The row level affects the output in the SXD file. For example, if you select row level **Output as blocks**, block elements will be included in the SXD file.
Create SXD	The path to the SXD file.

MessageIN Event configuration

The name of the MessageIN Event must be the same as **Event name** specified in the JDBC input connector. Since the Event does not use any SXD file in this case, you must specify a pattern for the Event (open the Event tool and enter the **Pattern**). For example, if **Event name** is `CustomerData` you can also use `CustomerData` as **Pattern**.

JDBC output connector

The JDBC output connector is used to insert or update data in relational databases. In the connector configuration you specify the appropriate JDBC driver and Connection URL settings to connect to the database, and the SQL query to execute on the selected database.

XMLOUT Process

The output sent via the JDBC output connector is configured using an XMLOUT Process. The XML structure is shown in the example below.

```
<?xml version="1.0"?>
<document>
  <job>
    <event>
      <block>
        <field name="column_1">value_1</field>
        <field name="column_2">value_2</field>
        ....
        <field name="column_N">value_N</field>
      </block>
    </event>
  </job>
</document>
```

Figure 1 XML structure for output delivered via a JDBC output connector.

XML element	Description
`<event>`	The `<event>` element contains all the rows to be added to the database table.
`<block>`	Each row in the database table is represented by a `<block>` element.
`<field>`	Each column in a table row is represented by a `<field>` element. • The `name` attribute represents the column (column_1, column_2, etc.). The attribute value must be the same as the column name in the database. • The field value represents the value to add to the corresponding column (value_1, value_2, etc.). The field value is retrieved from a field or variable defined in the Event.

JDBC output connector settings

Connector type

Select `Java`.

Java class

Select `JDBCOutConnector`. When you click the button attached to this field, all available Java output connector classes are displayed. To activate the JDBC output connector settings you must select the java class `JDBCOutConnector`.

JDBC driver

The JDBC driver to use to connect to the database. You can select the following drivers from the drop-down list:

To use this driver...	Select this option
JDBC-ODBC bridge	`sun.jdbc.odbc.JdbcOdbcDriver`
DataDirect SQL Server driver	`com.streamserve.www.jdbc.sqlserver.SQLServerDriver`
DataDirect Oracle driver	`com.streamserve.www.jdbc.oracle.OracleDriver`

If you are using other drivers than the above, you can enter the corresponding driver expression in this field.

Connection URL

The connection URL to connect to the database. You can select and edit the following Connection URLs from the drop-down list:

If you are using this driver...	Select and edit this option

JDBC-ODBC bridge	**Option:**
	`jdbc:odbc:DB`
	Example:
	`jdbc:odbc:myDatabase`

DataDirect SQL Server driver	**Option:**
	`jdbc:streamserve:sqlserver://server[:port];databaseName=DB`
	Example:
	`jdbc:streamserve:sqlserver://localhost:1433;databaseName=myDatabase`

DataDirect Oracle driver	**Option:**
	`jdbc:streamserve:oracle://server[:port];SID=DB`
	Example:
	`jdbc:streamserve:oracle://localhost:1433;SID=myDatabase`

If you are using other drivers than the above, you can enter the corresponding Connection URL in this field.

User name

A user name to access the database. Overrides any other user names specified in the Connection URL.

Password

A password to access the database. Overrides any other passwords specified in the Connection URL.

SQL file

You can create an SQL file where you enter the query to submit to the database. This query overrides any query specified in the **SQL command** field described below.

If you use the **SQL file** option to specify the query, you can create more complex queries using several lines of statements.

Example 46 Query in SQL file

```
INSERT INTO myTable *
```

SQL command

You can enter a one-line query in the SQL command field.

Example 47 Query in SQL command

```
INSERT INTO myTable *
```

License Password

Password for the JDBC driver (if required). If you are using any of the DataDirect JDBC drivers included in the StreamServe installation you should leave this field blank.

JMS connectors

The *JMS Queue input connector* enables StreamServer to receive messages from a named queue in a point-to-point messaging system.

The *JMS Queue output connector* enables StreamServer to deliver messages to a named queue in a point-to-point messaging system.

The *JMS Subscribe input connector* enables StreamServer to receive messages from a named topic in a publish-subscribe messaging system.

The *JMS Publish output connector* enables StreamServer to deliver messages to a named topic in a publish-subscribe messaging system.

The *JMS Response output connector* enables StreamServer to send a response to a received message.

JMS Queue input connector

This connector enables StreamServer to receive messages from a named queue in a point-to-point messaging system. For more information about JMS, see http://java.sun.com/products/jms/.

Connector settings	
Setting	**Description**
Context	The JNDI context. If you leave this empty, the initial context will be used.
Queue Factory	The JNDI lookup name for the queue connection factory. A JMS client uses a connection factory to create a connection to the JMS provider. This setting is mandatory.
User Name	A user name to access the JMS provider.
Password	A password to access the JMS provider.
Queue	The JNDI lookup name for the queue. This setting is mandatory.
Selector	You can use a message selector to apply filters to the received messages. A message selector is a string that contains an expression. The syntax of the expression is based on a subset of the SQL92 conditional expression syntax. **Example:** `Type = 'Invoice' OR Type = 'Order'` This message selector selects messages where Type is set to either Invoice or Order.

Connector settings	
Setting	**Description**
Polling interval	Opens the Scheduler Configuration dialog where you specify when and how often to retrieve messages. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.
Time-out	Specifies a time-out (seconds) for the connector. StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

JMS Subscribe input connector

This connector enables StreamServer to receive messages from a named topic in a publish-subscribe messaging system. For more information about JMS, see http://java.sun.com/products/jms/.

Connector settings	
Setting	**Description**
Context	The JNDI context. If you leave this empty, the initial context will be used.
Topic Factory	The JNDI lookup name for the topic connection factory. A JMS client uses a connection factory to create a connection to the JMS provider. This setting is mandatory.
User Name	A user name to access the JMS provider.
Password	A password to access the JMS provider.
Topic	The JNDI lookup name for the topic. This setting is mandatory.
Selector	You can use a message selector to apply filters to the received messages. A message selector is a string that contains an expression. The syntax of the expression is based on a subset of the SQL92 conditional expression syntax. **Example:** `Type = 'Invoice' OR Type = 'Order'` This message selector selects messages where Type is set to either Invoice or Order.
Subscriber Id	A subscriber ID for the subscription service set up via this connector. The subscriber ID enables the JMS provider to postpone sending messages to this subscriber if the connection between StreamServer and the JMS provider is lost. No messages will be lost during the time the connection is down.
Unsubscribe on close	Unsubscribe when StreamServer is shut down. You must specify a **Subscriber Id** to enable this feature.
Polling interval	Opens the Scheduler Configuration dialog where you specify when and how often to retrieve messages. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.
Time-out	Specifies a time-out (seconds) for the connector. StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

JMS Publish output connector

This connector enables StreamServer to deliver messages to a named topic in a publish-subscribe messaging system. For more information about JMS, see http://java.sun.com/products/jms/.

Connector settings	
Setting	**Description**
Context	The JNDI context. If you leave this empty, the initial context will be used.
Topic Factory	The JNDI lookup name for the topic connection factory. A JMS client uses a connection factory to create a connection to the JMS provider. This setting is mandatory.
User Name	A user name to access the JMS provider.
Password	A password to access the JMS provider.
Topic	The JNDI lookup name for the topic. This setting is mandatory.
Custom Properties	Custom message properties. Custom message properties can be used for selecting messages, or for providing information about contents or message types. **Example:** Set a custom header to `"color='blue'"`
Message Format	• **BytesMessage** – Send the message as bytes. The recipient must know how to interpret the information. Use this option if you specify a driver for the connector. • **TextMessage** – Send the message as text. Use this option when sending XML or text formatted output.
Delivery Mode	The message reliability level. You can select a delivery mode to suit your requirements for guarantee of delivery and system performance. • **Default** – Keep the settings defined in the deployment process. • **Persistent** – Output will be delivered even if power is lost. • **Non persistent** – Output will be delivered faster, but delivery is not guaranteed if power is lost.
Priority	The JMS priority, where 0 is the lowest priority, and 9 is the highest priority. If you do not specify a priority level, the default level is 4. A JMS provider tries to deliver messages with high before messages with low priority, but will not necessarily deliver the messages in the order of priority.
Time to live	An expiration time (milliseconds) for the messages. By default, a message will live forever.

Connector settings	
Setting	**Description**
Type	A `JMSType` header field. You can, for example, specify the type of message when using the message format `BytesMessage`. Receivers can use the `JMSType` header field to determine whether or not they can interpret this kind of message.
Reply To	The JNDI lookup name of the queue or topic to which the message receiver should send a response message. Specify this only if you expect the message receiver to reply to the delivered message.
Correlation Id	A correlation id for the message. Correlation IDs can be used to link one message to another, for example to link a response message to the corresponding request message. It is possible to fetch the correlation ID for incoming JMS messages using the scripting function *GetExtJobId*.

JMS Queue output connector

This connector enables StreamServer to deliver messages to a named queue in a point-to-point messaging system. For more information about JMS, see http://java.sun.com/products/jms/.

Connector settings	
Setting	**Description**
Context	The JNDI context. If you leave this empty, the initial context will be used.
Queue Factory	The JNDI lookup name for the queue connection factory. A JMS client uses a connection factory to create a connection to the JMS provider. This setting is mandatory.
User Name	A user name to access the JMS provider.
Password	A password to access the JMS provider.
Queue	The JNDI lookup name for the queue. This setting is mandatory.
Custom Properties	Custom message properties. Custom message properties can be used for selecting messages, or for providing information about contents or message types. **Example:** Set a custom header to `"color='blue'"`
Message Format	• **BytesMessage** – Send the message as bytes. The recipient must know how to interpret the information. Use this option if you specify a driver for the connector. • **TextMessage** – Send the message as text. Use this option when sending XML or text formatted output.
Delivery Mode	The message reliability level. You can select a delivery mode to suit your requirements for guarantee of delivery and system performance. • **Default** – Keep the settings defined in the deployment process. • **Persistent** – Output will be delivered even if power is lost. • **Non persistent** – Output will be delivered faster, but delivery is not guaranteed if power is lost.
Priority	The JMS priority, where 0 is the lowest priority, and 9 is the highest priority. If you do not specify a priority level, the default level is 4. A JMS provider tries to deliver messages with high before messages with low priority, but will not necessarily deliver the messages in the order of priority.
Time to live	An expiration time (milliseconds) for the messages. By default, a message will live forever.

Connector settings	
Setting	**Description**
Type	A `JMSType` header field. You can, for example, specify the type of message when using the message format `BytesMessage`. Receivers can use the `JMSType` header field to determine whether or not they can interpret this kind of message.
Reply To	The JNDI lookup name of the queue or topic to which the message receiver should send a response message. Specify this only if you expect the message receiver to reply to the delivered message.
Correlation Id	A correlation id for the message. Correlation IDs can be used to link one message to another, for example to link a response message to the corresponding request message. It is possible to fetch the correlation ID for incoming JMS messages using the scripting function *GetExtJobId*..

JMS Response output connector

This connector enables StreamServer to send a response to a received message. The received message must include information about which queue or topic to send the response to. For more information about JMS, see http://java.sun.com/products/jms/.

Connector settings	
Setting	**Description**
Queue Factory	The JNDI lookup name for the queue connection factory.
Queue User Name	A user name to access the (queue) JMS provider.
Queue Password	A password to access the (queue) JMS provider.
Topic Factory	The JNDI lookup name for the topic connection factory.
Topic User Name	A user name to access the (topic) JMS provider.
Topic Password	A password to access the (topic) JMS provider.
Custom Properties	Custom message properties. Custom message properties can be used for selecting messages, or for providing information about contents or message types. **Example:** Set a custom header to `"color='blue'"`
Message Format	• **BytesMessage** – Send the message as bytes. The recipient must know how to interpret the information. Use this option if you specify a driver for the connector. • **TextMessage** – Send the message as text. Use this option when sending XML or text formatted output.
Delivery Mode	The message reliability level. You can select a delivery mode to suit your requirements for guarantee of delivery and system performance. • **Default** – Keep the settings defined in the deployment process. • **Persistent** – Output will be delivered even if power is lost. • **Non persistent** – Output will be delivered faster, but delivery is not guaranteed if power is lost.
Priority	The JMS priority, where 0 is the lowest priority, and 9 is the highest priority. If you do not specify a priority level, the default level is 4. A JMS provider tries to deliver messages with high before messages with low priority, but will not necessarily deliver the messages in the order of priority.
Time to live	An expiration time (milliseconds) for the messages. By default, a message will live forever.

Connector settings	
Setting	**Description**
Type	A `JMSType` header field. You can, for example, specify the type of message when using the message format `BytesMessage`. Receivers can use the `JMSType` header field to determine whether or not they can interpret this kind of message.
Reply To	The JNDI lookup name of the queue or topic to which the message receiver should send a response message. Specify this only if you expect the message receiver to reply to the delivered message.
Correlation Id	A correlation id for the message. Correlation IDs can be used to link one message to another, for example to link a response message to the corresponding request message. It is possible to fetch the correlation ID for incoming JMS messages using the scripting function *GetExtJobId*.

Job Resource output connector

This connector enables the StreamServer to temporarily store data as a job resource that will be used later on.

Comments
You can use the EMF Print processor to convert EMF files into LXF overlays. The LXF overlay is sent via a Job Resource output connector to a temporary storage. The LXF overlay can then be called from a PageOUT Process and be added to the processed output. You must also use the Job Resource connector in an HTTPIN-HTTP response scenario, where the client expects the response to be presented in a web browser via an ActiveX plug-in. See *Responding with processed data via a Job Resource connector* on page 72.

The connector settings are described below.

Name

A name used together with scripting functions to identify a specific job resource.

Content-type

The content-type of the job resource. Used only with HTTP(S) input and HTTP Response output connectors.

Example 48 *Content-type*

```
text/html; charset="ascii"
```

Custom header

A user defined header that will be added to the job resource. Used only with HTTP(S) input and HTTP Response output connectors. Use the following syntax:

`Name:Value`

You can separate `Name:Value` pairs with `<0d><0a>`.

Expire by time

Specify whether to remove the job resource from the temporary storage.

- **No** – Do not remove the job resource.
- **Yes** – Remove the job resource from the temporary storage after the time specified by **Duration** and **Time unit** below.
- **Renew** – Same as Yes, but the counter is restarted each time the job resource is accessed.

Duration and Time unit

The time limit for **Expire by time** above.

Expire by access

Specify whether to limit the number of times a resource can be accessed before it is removed from the temporary storage. All resource operations within the same job are considered as one access.

- **No** – Do not remove the job resource.
- **Yes** – Remove the job resource from the temporary storage when Access Count specified below is reached.

Access count

The number of times a resource can be accessed when **Expire by access** above is set to Yes.

Expire with job

Specify whether to remove the job resource from the temporary storage when the job is completed.

Related scripting functions

The following script functions are related to this connector:

- *OutputLXFJobResource*
- *DeleteJobResource*
- *GetJobResourceIndex*

LiveCycle output connector

StreamServer can use a LiveCycle output connector to invoke LiveCycle processes that are deployed within Adobe LiveCycle ES and exposed through web services. These web services can be used to integrate LiveCycle processes into the StreamServer pipeline when processing documents.

The connector settings are described below.

Host

The host name or IP address of the server hosting LiveCycle ES. For example:

```
localhost
```

Port

The port used by the LiveCycle ES server. For example:

```
8080
```

Web service name

The name (case sensitive) of the service to invoke. This name must be the same as the corresponding process created in LiveCycle Workbench ES.

User name

User name to connect to the server hosting LiveCycle ES. Used in case of basic HTTP authentication.

Password

Password to connect to the server hosting LiveCycle ES. Used in case of basic HTTP authentication.

Enable asynchronous communication

Yes	Make asynchronous calls to the service. This option is used when invoking long-lived LiveCycle services.
No	Make synchronous calls to the service. This option is used when invoking short-lived LiveCycle services.

Asynchronous poll interval

Only used together with asynchronous calls. This is the interval (milliseconds) used to check for a response to the invocation request.

Root certificate for SSL communication

The root certificate used when HTTPS is used as web service protocol (secure communication). The certificate must be available from a resource set connected to the Platform.

Custom options

A list of custom keys (key-value pairs) to include in the invocation request.

To be able to handle custom keys, the service must have a variable named `optionsMap` of the type `map`. All custom keys defined here will be added to the `optionsMap` variable in the invoked service.

The values provided can be extracted in the receiving LiveCycle process by using an XPath expression in the LiveCycle process.

Examples of custom keys are passwords for creating password encrypted PDF files. For example:

Key: `pdfpassword`

Value: `encrypted`

LiveLink ECM in R3 output connector

This connector writes output to a LiveLink ECM transfer directory.

Comments
This connector was named IXOS Archive in R3 in previous releases. The name is now changed to LiveLink ECM due to ownership.

The connector settings are described below.

LiveLink ECM transfer directory

The path to the LiveLink ECM transfer directory.

Meta documents

Specifies whether to enable the use of meta documents.

Max. no of documents in meta

The maximum number of documents to be stored in each meta document directory. When the maximum number of documents have been added, a new document directory is created.

Fixed directory name

- **Yes** – The StreamServer adds all documents to the fixed directory (see **Directory Name** below).
- **No** – The StreamServer generates unique directories, with a maximum of 1000 documents in each directory.

Runtime specific connector settings

IXATTR file lines

The lines to add to the IXATTR control file.

Commands file lines

The lines to add to the Commands control file.

Autodetect Document Type

Select to let the document type for the output be set by the driver being used.

Document Type

Specify the document type manually.

Autodetect file extension

Select to let the file extension for the output be set by the document type specified.

File extension

Specify the file extension manually.

Directory Name

The name of the fixed directory.

Component Name

The name to be inserted in the Commands file.

Lotus Notes output connector

This connector writes output to a Lotus Notes database.

Comments
If you run the StreamServer and a Lotus Notes client version 6.x or 7.0 on a Windows 2003 Server, you must install Citrix to use the Lotus Notes output connector.

The connector settings are described below.

Notes Server

The Lotus Domino server to connect to.

Example 49 *Notes Server*

```
windows_server05/my_domino_server
```

where `windows_server05` is the computer hosting the Lotus Domino server.

Runtime specific connector settings

Notes Database

The Lotus Notes database on the Lotus Domino server to connect to. The Lotus Notes database stores the output from the StreamServer as an attachment to a Note.

The specified database must allow write access for the user specified in the Lotus ID file. Only a Lotus Domino administrator can enable read or write access to the Lotus Notes database.

If you connect to the database via a Lotus Domino server, without access to a Lotus Notes client installation, you must:

- Obtain a suitable Lotus ID file.
- Ensure that the Lotus ID file is recertified and contains a valid password.
- Rename the Lotus ID file to `ssdominouser.id`
- Store the Lotus ID file in the working directory of the StreamServer.

Ask your Lotus Domino administrator for assistance.

Example 50 Notes Database

```
custmsgs.nsf
```

Password

The password for unlocking the Lotus ID file which authenticates access to a Lotus Notes database. The StreamServer searches for the Lotus ID file named `ssdominouser.id` in the working directory of the StreamServer. If the Lotus ID file is not found, or if the password fails to unlock it, the StreamServer tries to use the previously used Lotus ID file that `ssnotes.ini` has a reference to.

Default Form

The Lotus Notes form to be used. To create a Lotus Notes form on the Lotus Domino server, you must use the Domino Designer client.

Filename

The name of the file created by the StreamServer and attached to the Note.

Note: The Temporary directory must be specified with a full path for the StreamServer instance running the Project.

Example 51 Filename

```
$docname.pdf
```

Compress

Select to compress the file attached to the Note when sending it to the Domino Server.

Indexes

In addition to the attached file produced by the StreamServer, you can add data about the attachments to the corresponding Note. This is useful if you want data about the attachment to be easily accessible by the Lotus Notes client or the Lotus Notes Fetch wizard, without having to open the attachment.

To add this data for a Note, you must enter the items that will be included in the Note and a variable for each item.

- **Index Name** – The Note item name.
- **Data Type** – The type of the item data.
- **Variable** – A StreamServe variable. The value of the variable is stored in the Note.

Custom connector settings

You can add custom keywords to the Lotus Notes connector. See the *Design Center* documentation for more information about how to add custom commands and keywords.

`lotusdbopenretrydelay <open_delay>`
The time to wait in milliseconds between attempts to open the Lotus Notes database. For example `lotusdbopenretrydelay 2000`. The default value is 1000.

`lotusdbopenmaxretries <open_retries>`
The number of times to attempt to open the Lotus Notes database. For example `lotusdbopenmaxretries 5`. The default value is 3.

`lotusconnretrydelay <send_delay>`
The time to wait in milliseconds between attempts to send a Lotus Note to the Lotus Domino server via a Lotus Notes output connector. For example `lotusconnretrydelay 10000`. The default value is 5000.

`lotusconnmaxretries <send_retries>`
The number of times to attempt to send a Note to the Lotus Domino server via a Lotus Notes output connector. For example `lotusconnmaxretries 10`. The default value is 1.

MSMQ connectors

The *MSMQ input connector* and *MSMQ output connector* are used for integrating the StreamServer to Microsoft message queuing environments.

MSMQ input connector

This connector enables the StreamServer to integrate with Microsoft message queuing environments.

Comments
Segmented messages If the received messages are segmented, the application sending the messages must synchronize the messages as follows: • The first message in the sequence must have the `appspecific` property set to the number of messages that will arrive in this sequence of segmented messages (the first message is excluded). • The following messages must have this number decreased by one and the last message in the sequence must have the `appspecific` property set to 0. • All messages, except the first one, must have their correlation ID property set to the message ID of the first message. **Private and public message queues** Public message queues are published in the Active Directory. Private message queues are not published in the Active Directory. They are accessible only by Message Queuing applications that know the full path name or format name of the message queue.

The connector settings are described below.

Queue Manager

The server of the MS Message Queue.

Queue Name

The MS Message Queue from which to retrieve data. This MS Message Queue must belong to the Queue Manager specified above.

Private

Select to define the MS Message Queue as a private queue. Private message queues can be accessed even if the queue manager is off-line.

Transaction Type

- **MQ_NO_TRANSACTION** – All messages will be read without transaction support.
- **MQ_INTERNAL_TRANSACTION** – Messages that are not successfully processed will be rolled-back, and the StreamServer will try to process them again. Transactional messages are only read (removed) from a connector if and when the transaction is committed. Otherwise they are returned to the connector and can be read during a subsequent transaction.

Polling interval

Opens the Scheduler Configuration dialog where you specify when and how often to retrieve data. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.

Time-out

Specifies a time-out (seconds) for the connector. The StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, the StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

MSMQ output connector

This connector enables the StreamServer to integrate with Microsoft message queuing environments.

Comments
Public message queues are published in the Active Directory. Private message queues are not published in the Active Directory. They are accessible only by Message Queuing applications that know the full path name or format name of the message queue.

Queue Manager

The server of the MS Message Queue.

Queue Name

The MS Message Queue to which to transfer data. This MS Message Queue must belong to the Queue Manager specified above.

Private

Select to define the MS Message Queue as a private queue. Private message queues can be accessed even if the queue manager is off-line.

Max Message Size

The maximum size (bytes) for an MSMQ message. The maximum value allowed is 4194304. If the data sent from the StreamServer exceeds the maximum size, it will be split into several messages. These messages will be synchronized as follows:

- The first message in the sequence will have the **appspecific** property set to the number of messages that will arrive in this sequence of segmented messages (the first message is excluded).
- The following messages will have this number decreased by one and the last message in the sequence will have the **appspecific** property set to 0.
- All messages, except the first one, will have their correlation ID property set to the message ID of the first message.

Runtime specific connector settings

Message Delivery

- **Express** – The message is not stored on disk, and cannot be recovered in case of failure.
- **Persistent** – The message is stored on disk, and can be recovered in case of failure.

Message Priority

The priority of the message.

Message Encryption

Select to enable message encryption. The encryption uses symmetric key and asymmetric (public/private) key encryption algorithms. Encryption slows down transfer of messages.

Message Journaling

Specifies the level of journaling. Journals provide audit trails.

- **MQMSG_JOURNAL_NONE** – No journaling.
- **MQMSG_JOURNAL** – Journaling enabled.
- **MQMSG_DEADLETTER** – If the message is not delivered or read in time, it will be stored in a deadletter queue.
- **MQMSG_JOURNAL|MQMSG_DEADLETTER** – Both journaling and deadletter functionality enabled.

Max Time to Reach Queue

The maximum time (seconds) for the message to reach the destination queue. If the message does not reach the destination queue within this time, and you have enabled one of the dead-letter options, the message will be moved to the dead-letter queue.

Max Time to Receive

The maximum time (seconds) to wait for the message to be read. If the message is not read within the specified time, and you have enabled one of the dead-letter options, the message will be moved to the dead-letter queue.

Message Tracing

Specifies the level of tracking. To enable message tracing, message route tracking must be enabled by defining a report queue for the sending party.

- **Tracing Off** – No tracing of the progress of the message.
- **Tracing On** – Progress of the message is traced.

Transaction Type

Specifies the transaction type. Transactional messages are either sent together and in the order they were sent (a committed transaction), or not sent at all (an aborted transaction). The receiving MSMQ queue must support transactions.

- **MQ_NO_TRANSACTION** – The message will be sent without transactional support.
- **MQ_INTERNAL_TRANSACTION** – The message will be sent within a transaction.

Message Label

A descriptive text to include in the message.

Null Connector output connector

This connector enables the StreamServer to send output to a dummy connector that does not direct output to any file or printer. There are no connector settings for this type of connector.

Pipe connectors

The *Named Pipe input connector* is used for retrieving input from a named pipe.

The *Pipe output connector* (UNIX only) is used for sending output to a named pipe.

Named Pipe input connector

This connector retrieves input from a named pipe. The connector settings are described below.

Named Pipe

The named pipe.

Schedule

Opens the Scheduler Configuration dialog where you specify when and how often to try to retrieve data. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.

Time-out

Specifies a time-out for the connector. The StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, the StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

Pipe output connector

This connector sends output to a named pipe (UNIX only). The connector settings are described below.

Pipe

The command to execute.

Example 52 Pipe

```
cat
```

Service Call output connector

This connector is used by a StreamServer application in the following scenarios:

- In the SAP Business Processes solution, this connector is used to send IDoc data to a SAP system.
- To call another StreamServer application that has a Service Request input connector.

SAP Business Process solution

In the SAP IDoc inbound - StreamServe IDoc outbound scenario, a StreamServer application receives data from any data source, and processes the data as an XMLOUT Process. The Service Call output connector stores the XML in the runtime repository, where the IDoc Client picks it up for converting and sending to SAP.

For more information about using the Service Call output connector in the Business Processes solution, see the *StreamServe Connect for SAP - Delivery Manager* documentation.

StreamServer application with a Service Request input connector

In this scenario, the output connector calls another StreamServer application that is exposed as a service using a Service Request input connector.

Depending on how you configure the output connector, the receiving StreamServer application can return the status of the job it processes.

Connector settings

Service Type

IDocClient – Used to send IDoc data to SAP.

General – Used to call a StreamServer application with a Service Request input connector.

Service name

In the SAP IDoc scenario, this is the Service name that you have specified on the IDoc Client in the RFC Gateway. On the IDoc client, the Service name setting is found under the StreamServe Connection settings.

When calling another StreamServer application, this is the service name of the receiving Service Request input connector.

Timeout

When calling another StreamServer application with a Service Request input connector, this setting determines the following:

- Whether the output connector waits for the receiving StreamServer application to return the status of the job it processes.
- How long to wait for a response.

Setting	Description
-1	The connector waits indefinitely for the receiving StreamServer application to return the job status before it marks the output job as successful or failed.
0	The output connector marks the output job as successful/failed when the receiving Service Request input connector places the job in an input queue.
time (milliseconds)	The time the connector waits for the receiving StreamServer application to return the job status. If the status is not returned within this time period, the output job is marked as successful.

Service Channel (HTTP) connectors

The *Service Channel (HTTP) input connector* is used for publishing services in a Service Broker. Clients can access the service via HTTP.

The *Service Channel Response (HTTP) output connector* is used together with a Service Channel (HTTP) input connector. It sends the processed output back to the client.

The *Service Channel Submit (HTTP) output connector* is used for invoking services via a Service Broker.

Service Channel (HTTP) input connector

This connector publishes services in a Service Broker. Clients can access the service via HTTP. See the *Service Broker* documentation for more information. The connector settings are described below.

Service description

The name of the service published in the Service Broker. This name will be used in the client requests when requesting this specific service. You can define several names, separated by comma, for the same service.

Example 53 *Service description*

```
PDF_OUT
```

Version

The version of the service. There can be several versions of the same service. A client can request a specific version of a service. If the client does not request a specific version, it will get the highest available version of the service.

Service Channel Response (HTTP) output connector

This connector is used together with a *Service Channel (HTTP) input connector*. It sends the processed output, via a Service Broker, back to the client. See the *Service Broker* documentation for more information. The connector settings are described below.

Content-type

The content-type of the output.

Example 54 Content-type

```
text/html
```

Service Channel Submit (HTTP) output connector

This connector invokes services via a Service Broker. See the *Service Broker* documentation for more information. The connector settings are described below.

Service Broker host

The Service Broker host.

Port

The Service Broker port.

Service description

The name of the requested service.

Example 55 *Service description*

```
PDF_OUT
```

Server ID

The server ID of a specific StreamServer. Specify this only if you want to select a specific StreamServer. If you do not specify a Server ID, the Service Broker will select the appropriate StreamServer.

Example 56 *Server ID*

```
124.234.7.31\StreamServe4
```

Version

The version of the service. If you leave this empty, the highest registered version of the service is invoked.

Content-type

The content-type of the submitted output.

Example 57 *Content-type*

```
application/pdf
```

Command

- **SendDocument** – Send the output via the Service Broker to a StreamServer. The StreamServer will process the data, and send the processed output to a printer, fax, etc..

- **SendReceiveDocument** – Send the output via the Service Broker to a StreamServer. The StreamServer will process the data, and send the processed output back in the response.
- **AddDocument** – Send the output to the Service Broker. The Service Broker will add this document to a response that has the same **Doc Group ID** or Strs ID as this document.

Response service channel

The service name of the Service Channel (HTTP) input connector that will receive the response. Used together with the command **SendReceiveDocument**.

Connection time-out

The maximum time (milliseconds) to wait to connect to the Service Broker.

Communication time-out

The maximum time (milliseconds) to wait before aborting transfer/reception of data.

Document group ID

The response to add documents to. The instance that issued the `SendReceiveDocument` request is the instance that will receive the response. The instance that issued the `AddDocument` request is the instance that will add the document to the above response. The document will be added to the response if either:

- Both instances use the same **Document Group ID**.
- No **Document Group ID** is specified, and the response and document to add have the same Strs ID.

LastDoc

Defines the current document as the last document to add. Used together with the command **AddDocument**.

Service Request input connector

This connector is used by a StreamServer application in the following scenarios:

- To retrieve input from Adobe LiveCycle ES processes.
- To retrieve documents from Collector.
- To retrieve IDoc data from SAP.
- To retrieve data from SAP over the XOM interface.

The connector exposes the StreamServer application as a web service to the client.

Note: The connector must be connected to an input queue.

Adobe LiveCycle ES scenario

In an Adobe LiveCycle ES scenario, any type of document can be sent via a Service Request connector to the StreamServer application for further processing. LiveCycle ES invokes the StreamServer application by issuing a service request, and delivers the input data in the request.

The service name specified on the Service Request connector is the name of the web service LiveCycle ES must call to invoke the StreamServer application.

Collector scenario

In a Collector scenario, documents are forwarded from Collector via the Service Request connector to the StreamServer application for reprocessing. Collector invokes the StreamServer application by issuing a service request, and delivers the input data in the request.

Collector can call three predefined web services to invoke the StreamServer application. All these web service options are available from a drop-down list in the Service Request connector configuration.

SAP IDoc scenario

In a SAP outbound - StreamServe inbound scenario, your SAP system sends IDoc data to the StreamServer via a Service Request input connector (for the ALE interface).

SAP XOM scenario

If you run the Delivery Manager Server/Client level of integration, you can use the Service Request input connector for receiving input from SAP.

Connector settings

Request type

- **Generic** – select this option if the client is Adobe LiveCycle ES.
- **Collector reprocess** – select this option if the client is Collector.
- **IDoc** – select this option to receive IDoc data.
- **XOM** – select this option to receive data from the Delivery Manager Sender.

Service name

Enter the name of the web service the client must call to invoke the StreamServer application.

Note: Not available if Request type is set to Collector reprocess.

Reprocess type

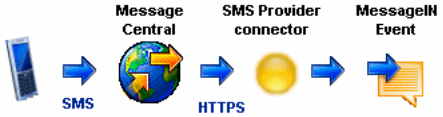
Available only if Request type is set to Collector reprocess.

- **Print** – select if the reprocessed documents should be printed.
- **Fax** – select if the reprocessed documents should be sent by fax.
- **Mail** – select if the reprocessed documents should be sent by email.

SMS connectors

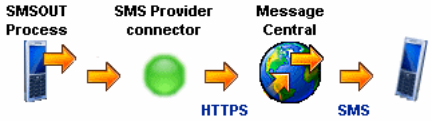
The StreamServer can send and receive SMS messages via the service “Mobil Text Företag”. This service is provided by the swedish operator Telia, and is only available in Sweden. The data between Telia’s Message Central and the StreamServer is transferred over HTTPS.

Receiving SMS



1. The Message Central receives an SMS message and sends it to the StreamServer over HTTPS.
2. The StreamServer receives the SMS message via an *SMS Provider input connector*.
3. The input is picked up by a MessageIN Event for further processing in the StreamServer.

Sending SMS



1. The StreamServer uses an SMSOUT Process to generate SMS output.
2. The SMS output is sent from the StreamServer to the message central via an *SMS Provider output connector*.
3. The Message Central sends the output to the recipients.

Sending and receiving SMS

You must do a number of things before you configure the StreamServer to send and receive SMS:

- You must have a subscription to Telia's service "Mobil Text Företag".
- You must get an account and password from the service provider.
- You must get a CA certificate from the service provider.
- The service provider must know which IP address and port the StreamServer will use for receiving SMS Messages.

Sending SMS

To be able to send SMS from the StreamServer via an external service provider you must:

- Configure an SMSOUT Process that generates the SMS messages.
- Configure an SMS Provider output connector that delivers the output to the service provider.

To configure the SMSOUT Process

Open the SMSOUT Process and enter the SMS message as text or variables.

To configure the SMS Provider output connector

Before you configure the connector you must import the service provider's CA certificate to a resource set connected to the Platform. Then you configure the *SMS Provider output connector* according to standard procedures.

Receiving SMS

To be able to receive SMS messages via an external service provider you must:

- Configure an SMS Provider input connector that receives the input from the service provider.
- Configure a MessageIN Event that handles the SMS input.

To configure the SMS Provider input connector

Configure the *SMS Provider input connector* according to standard procedures.

To configure the MessageIN Event for SMS

1. Import `SMS_Provider.sxd` to a resource set connected to the Message. This file is located in `...\Tools\System\data\sxd`.
2. Create a MessageIN Event and name it `SMS_Provider`.
3. Open the Event, browse to and select the resource `SMS_Provider.sxd`.
4. Click **Import Message**.
5. Save and close the Event.

SMS Provider input connector

This connector receives SMS messages.

Comments
The StreamServer uses the service "Mobil Text Företag" to receive SMS messages. This service is provided by the swedish operator Telia, and is only available in Sweden. The data between Telia's Message Central and the StreamServer is transferred over HTTPS.

The connector settings are described below.

Host

The IP address of the StreamServer host. Must be registered with Telia.

Port

The port to listen to for incoming SMS messages. Must be registered with Telia.

Event name

The name of the MessageIN Event that receives the input from the SMS Provider input connector.

Time-out (ms)

The maximum time (milliseconds) the connector will try to carry out the transaction (connect, send data, receive data) before canceling.

SMS Provider output connector

This connector sends SMS messages via the external service provider Telia.

Comments
The StreamServer uses the service "Mobil Text Företag" to send SMS messages. This service is provided by the swedish operator Telia, and is only available in Sweden. The data between Telia's Message Central and the StreamServer is transferred over HTTPS.

The connector settings are described below.

Account

The account number provided by Telia.

Password

The password (provided by Telia) for accessing the account.

Address

The service provider URL. Must be:

```
https://telemat.telia.com/aps/APSlet
```

CA Certificate

The CA certificate provided by telia. The CA certificate must be included in a resource set connected to the Platform.

Runtime specific connector settings

Recipients

The recipient's mobile phone number(s). If you enter several numbers, you must separate them with semi-colon `(;)`.

Example 58 Recipients

```
070 555 5555;070 555 5556
```

SMTP output connector

This connector sends output via an SMTP server.

Comments

The recipient of the output will in turn retrieve the information from the SMTP server. This is similar to communication via HTTP. The main difference is that the recipient does not have to be up and running when the output is sent out.
You must specify the recipients, normally one or more mail boxes, of the output sent to the SMTP server. You must use the *SetDestPath* scripting function to specify the recipients. For example:
`SetDestPath("sulti@max.com;qtr@mux.com");`
`SetDestPath($email);`

The connector settings are described below.

Mail server

The IP address or host name of the SMTP server.

Return Address

The email address to which the SMTP server will report errors.

SMTP output connector

SNMP trap output connector

This connector sends SNMP traps to Network Management Systems (NMS) that use SNMP v1.

Comments
SNMP trap information The SNMP trap can include the following type of information: • **Message** – a description of the message that is sent. • **JobSource** – a description of the source that triggered the message. • **Type** – must be one of the following: `Job`, `Event`, `Process`, `input connector`, `output connector`, or `other`. • **Status** – must be one of the following: `Started`, `Running`, `Warning`, `Success`, `Fail`, or `Unknown`. **StreamServe MIB** The content of the SNMP trap is defined by the StreamServe MIB (Management Information Base) in `...\Common\Data\MIBS\StreamServe-MIB.txt`. **StreamServer configuration** Configuring the StreamServer to create SNMP traps includes: • SNMP trap output connector configuration. • *Collection configuration*. • *Process configuration*.

The connector settings are described below.

Trap destination

The IP address or hostname of the NMS.

Trap destination IP port

The port on which the NMS receives SNMP traps.

Community string

The password to access the NMS.

Source description

A description of the source sending the notifications.

Collection configuration

The StreamServer creates SNMP traps based on notifications generated by the source application (ERP system etc.) or by the StreamServer itself.

To collect and extract notifications from a source application you must use the appropriate input connector and Event.

To collect and extract notifications from within the StreamServer you must use a Status Messenger input connector and a MessageIN Event. See *Generating status reports* (*Status Messenger* documentation) for more information.

Process configuration

When you use a Status Messenger input connector and a MessageIN Event to collect and extract notifications, you should use a MessageOUT Process to configure the output. See the *MessageOUT* documentation.

When you use other types of Events to collect and extract notifications, you can use an XMLOUT Process to configure the output. See the *XMLOUT* documentation. The XMLOUT configuration must contain the fields to include in the SNMP trap.

Example 59 XML structure used for sending SNMP traps

```
<?xml version=\"1.0\"?>
<strs-xml version="1.0">
    <job>
         <event name="Garlic">
            <block>
                 <field name="Message" path="">message</field>
                 <field name="JobSource" path="">source</field>
                 <field name="Type" path="">job</field>
                 <field name="Status" path="">fail</field>
            </block>
        </event>
    </job>
</strs-xml>
```

Spool output connector

This connector sends output to a printer.

To

The printer to send the output to.

Spool output connector

Standard input and Standard output connectors

The *Standard input connector* enables external applications to send data to the StreamServer via StdIn.

The *Standard output connector* enables external applications to receive output from the StreamServer via StdOut.

Standard input connector

This connector enables external applications to send data to the StreamServer via StdIn. There are no settings for this type of connector.

Standard output connector

This connector enables external applications to receive output from the StreamServer via StdOut. There are no settings for this type of connector.

StreamServe External Viewer output connector

This connector sends output to a Previewer. The Previewer reads the file extension and opens the output file in the corresponding application (`*.pdf` in Acrobat Reader, etc.). The Previewer recognizes the following formats: `pdf`, `ps`, `tif`, `dcx`, `html`, `xml`, `pcl`. It tries to open other formats as `*.txt`.

See *About Previewer* for more information about Previewer.

Example 60 Previewer

You have one default Process that you connect to an output connector, and one identical preview Process that you connect to a StreamServe External Viewer connector. An Event script determines which Process to use:

```
if($var = "Preview")
   callproc("PreviewProcess");
else
   callproc("DefaultProcess");
```

The connector settings are described below.

Host

The Previewer host.

Port

The port the Previewer listens to. The default port is `9343`.

StreamServe External Viewer output connector

TCP/IP connectors

The *TCP/IP input connector* enables external applications to sent input to the StreamServer over a TCP/IP socket.

The *TCP/IP output connector* is used for sending output to a TCP/IP address.

TCP/IP input connector

This connector enables external applications to send input to the StreamServer over a TCP/IP socket. You can use the custom startup argument `-startallinconnectors` to start connectors that are not connected to any Event.

The connector settings are described below.

Host

The StreamServer host.

Port

The port number on which to receive input.

TCP/IP output connector

This connector sends output to a TCP/IP address. The connector settings are described below.

Host

The receiving host.

Port

The port the host listens to. The default port is `9343`.

TOPCALL output connector

TOPCALL® is a registered trademark of TOPCALL International AG.

This connector enables the StreamServer to send fax and email output via TOPCALL.

Comments
A Stream Serve configuration for TOPCALL includes a MailOUT Process, a PageOUT Process, and a Topcall output connector. The PageOUT output is sent as a PDF or PCL attachment, via the MailOUT Process and the Topcall connector, to a designated directory. TOPCALL picks up the attachment and sends it via fax or email. See *MailOUT Process and Topcall connector* on page 30.

The connector settings are described below.

TOPCALL To Directory

The destination for address information.

TOPCALL Attachment Directory

The destination for the StreamServer output files.

Runtime specific connector settings

See your *TOPCALL* documentation for information.

TOPCALL output connector

WebSphere MQ connectors

The *WebSphere MQ input connector* enables StreamServer to receive messages from IBM WebSphere message queues.

The *WebSphere MQ output connector* enables StreamServer to deliver messages to IBM WebSphere message queues.

Running 64-bit StreamServer against 64-bit WebSphere MQ

To be able to run a 64-bit StreamServer against 64-bit WebSphere MQ (zSeries platform), you must point `LD_LIBRARY_PATH` directly to the 64-bit WebSphere MQ libraries.

You must set the path to the WebSphere MQ libraries before all other paths. By setting `LD_LIBRARY_PATH` before you start `bootloader.sh`, it will appear at the beginning of `STRS_LIBRARY_PATH`.

Example 61 *Setting LD_LIBRARY_PATH*

```
$export LD_LIBRARY_PATH=/opt/mqm/lib64/
```

WebSphere MQ input connector

This connector receives input via IBM WebSphere MQ message queues. Before you can use this connector, you must add `-webspheremq` as a custom argument. You do this in the Configure Platform Export dialog box in Design Center.

Connector settings	
Setting	**Description**
Queue manager	The queue manager to connect to.
Queue name	The name of the queue to use.
Channel name	The name of the client connection channel. The name must be the same as the name of the server connection channel defined on the WebSphere MQ server.
Channel protocol	The protocol to use for WebSphere MQ communication. You can select one of the following: • TCP/IP • NETBIOS • SPX • LU62
Channel server name	Host name or IP address of the server running the queue manager.
Schedule	Opens the Scheduler Configuration dialog where you specify when and how often to retrieve input. See *Scheduling actions* in the *Design Center* documentation for more information about scheduling.
Time-out	Specifies a time-out (seconds) for the connector. StreamServer uses job-end sequences in the input data to determine when all data in an input job is received. If there are no job-end sequences in the input data, StreamServer will not know when to stop waiting for more input. To prevent this from happening, you can define a time-out for the input connector.

Additional information

Specifying a client connection channel using environment variable

You can use the environment variable `MQSERVER` to specify a default client connection channel to a specific queue manager. This will be the default client connection channel for all WebSphere MQ input and output connectors. If you specify `MQSERVER` and also specify valid Channel settings for a physical layer in the GUI (**Channel name**, **Channel protocol**, and **Channel server name**), this physical layer will use the client connection channel specified by the GUI settings (valid GUI settings override `MQSERVER` definition).

Use the following syntax to specify a client connection channel:

On Windows:

```
SET MQSERVER=<ChannelName>/<TransportType>/<ConnectionName>
```

On UNIX:

```
export MQSERVER=<ChannelName>/<TransportType>/<ConnectionName>
```

ChannelName	The name of the client connection channel. The name must be the same as the name of the server connection channel defined on the WebSphere MQ server. The *ChannelName* must not include forward slash (/).
TransportType	The protocol to use for WebSphere MQ communication. You can use one of the following: • TCP • NETBIOS • SPX • LU62
ConnectionName	Host name or IP address of the server running the queue manager.

Example 62 Specifying client channels on Windows.

```
SET MQSERVER=CHAN1/TCP/MCID66499
SET MQSERVER=CHAN2/TCP/9.20.4.56(1456)
SET MQSERVER=CHAN3/NETBIOS/BOX643
```

Example 63 Specifying client channels on UNIX.

```
export MQSERVER=CHAN1/TCP/'MCID66499'
export MQSERVER=CHAN2/TCP/'9.20.4.56'
export MQSERVER=CHAN3/LU62/BOX99
```

Groups and Segments

StreamServer and the application that sends data to StreamServer must handle group and segment properties in the same way. They must both read groups and segments using `MQGMO_LOGICAL_ORDER` to ensure that all messages are received in the same order as they were sent.

StreamServer uses `MQGMO_SYNCPOINT` to make sure the message remains in the queue if StreamServer fails to read or process the message. StreamServer will retry to read the message until it succeeds to do so. If it continuously fails to read the message, you must remove the message from the queue.

If a group of messages is sent, `MQGMO_ALL_MSGS_AVAILABLE` and `MQGMO_WAIT` are used to ensure that StreamServer waits until all messages in the group are available. StreamServer will also wait until all segments of a message are available.

WebSphere MQ attributes

You can use the scripting function *GetConnectorValue* to fetch WebSphere MQ attributes.

Attribute	**GetConnectorValue("<*argument*>")**
`GroupId`	`GetConnectorValue("IBMMQGroupId")`
`MessageId`	`GetConnectorValue("IBMMQMsgId")` `GetConnectorValue("IBMMQMsgIdBase64")`
`CorrelationId`	`GetConnectorValue("IBMMQCorrelId")` `GetConnectorValue("IBMMQCorrelIdBase64")`
`Reply-To-Queue`	`GetConnectorValue("IBMMQReplyToQ")`
`Reply-To-Queue-Manager`	`GetConnectorValue("IBMMQReplyToQMgr")`
`Message type`	`GetConnectorValue("IBMMQMsgType")`
`Report options`	`GetConnectorValue("IBMMQReport")`

WebSphere MQ output connector

This connector delivers output via IBM WebSphere MQ message queues. Before you can use this connector, you must add `-webspheremq` as a custom argument. You do this in the Configure Platform Export dialog box in Design Center.

Connector settings	
Setting	**Description**
Queue manager	The queue manager to connect to.
Queue name	The name of the queue to use.
Message persistence	• **AS_Q_DEF** – use the message persistency specified for the message queue. • **Yes** – write the message to disk. The message will be recoverable in case of system failure. • **No** – process the message in memory. This will improve performance at the expense of security.
Max. message size	The maximum message size StreamServer can handle without message segmentation. Default is 4Mb.
Allow queue manager segmentation	Enables the Queue Manager to segment the message. This may be necessary if the message is routed between different platforms, and the message exceeds the maximum message size on a platform.
Allow application segmentation	• **Yes** – if the message exceeds the maximum message size, it is divided into several smaller messages. • **No** – if the message exceeds the maximum message size, no data is sent to the message queue. An error message is written to the log.
Message format	• **String** – enable the Queue Manager to convert the message data to the character format used on the platform that receives the message. • **None** – disable the message data conversion. If a code page is specified, the data will be converted to this code page regardless of the Message Format setting.
Channel name	The name of the client connection channel. The name must be the same as the name of the server connection channel defined on the WebSphere MQ server.

Connector settings	
Setting	**Description**
Channel protocol	The protocol to use for WebSphere MQ communication. You can select one of the following: • TCP/IP • NETBIOS • SPX • LU62
Channel server name	Host name or IP address of the server running the queue manager.

Runtime specific connector settings	
Setting	**Description**
Message type	• **Request** – Used if an answer to the message is expected. You must also specify `ReplyToQ` and `ReplyToQMgr`. See *Custom connector settings* below. • **Reply** – Used if the message is a reply to a request message. • **Datagram** – Used if no answer to the message is expected. • **Report** – Used if you want the receiving application or the Queue Manager to report the message status to StreamServer. You must also specify `Report Option`. See *Custom connector settings* below.
Message ID	A message ID. Enables sending and receiving applications to correlate messages. Maximum length is 24 characters.
Correlation ID	A correlation ID. Specify a correlation ID for example if you are sending a reply message, and want the receiver to know which message you reply to. Use the Message Id from the request as your correlation ID. The maximum length is determined by the environment variable `MQ_CORREL_ID_LENGTH`. The recommended value is 24.

Custom connector settings

You can use the `QueueOption` keyword with the appropriate parameter to configure special runtime parameters. See *Using custom commands and keywords* in the *Design Center* documentation for information on how to add custom commands and keywords.

Parameter	Description
Report Option	Refer to the WebSphere MQ documentation for information about available Report Options. Must be long integer format (hexadecimal or decimal). For example: `QueueOption "Report Option" "0x101c000"`
ReplyToQ	The name of queue to reply to if a request message is sent or if any report options are set. For example: `QueueOption "ReplyToQ" "pdfQ"`
ReplyToQMgr	The name of the queue manager where the reply-to queue is hosted. For example: `QueueOption "ReplyToQMgr" "QM_Strs"`

Additional information

Specifying a client connection channel using environment variable

You can use the environment variable `MQSERVER` to specify a default client connection channel to a specific queue manager. This will be the default client connection channel for all WebSphere MQ input and output connectors. If you specify `MQSERVER` and also specify valid Channel settings for a physical layer in the GUI (**Channel name**, **Channel protocol**, and **Channel server name**), this physical layer will use the client connection channel specified by the GUI settings (valid GUI settings override `MQSERVER` definition).

Use the following syntax to specify a client connection channel:

On Windows:

`SET MQSERVER=<ChannelName>/<TransportType>/<ConnectionName>`

On UNIX:

`export MQSERVER=<ChannelName>/<TransportType>/<ConnectionName>`

ChannelName	The name of the client connection channel. The name must be the same as the name of the server connection channel defined on the WebSphere MQ server. The *ChannelName* must not include forward slash (/).

TransportType	The protocol to use for WebSphere MQ communication. You can use one of the following: • TCP • NETBIOS • SPX • LU62
ConnectionName	Host name or IP address of the server running the queue manager.

Example 64 Specifying client channels on Windows.

```
SET MQSERVER=CHAN1/TCP/MCID66499
SET MQSERVER=CHAN2/TCP/9.20.4.56(1456)
SET MQSERVER=CHAN3/NETBIOS/BOX643
```

Example 65 Specifying client channels on UNIX.

```
export MQSERVER=CHAN1/TCP/'MCID66499'
export MQSERVER=CHAN2/TCP/'9.20.4.56'
export MQSERVER=CHAN3/LU62/BOX99
```