StreamServe Persuasion SP5 Document Broker Plus

User Guide

Rev A

Contents

About Document Broker Plus

Document Broker Plus enables StreamServer applications to store documents in a common repository. A StreamServer application can retrieve the stored documents, and deliver the output via the appropriate output channels.

Variables and metadata can be stored with the documents, and be used by the retrieving StreamServer application for various purposes such as scripting, sorting, enveloping, etc.

The document storage (Post-processing storage) used by Document Broker Plus is a runtime repository. This means all documents are stored in a runtime repository, and no additional database for document storage is required

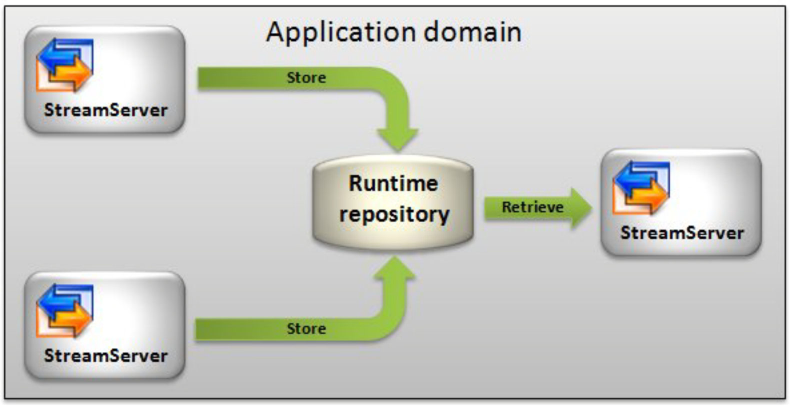


Figure 1 Runtime repository used as document storage

Storing and retrieving documents

Several StreamServer applications can store documents in the same runtime repository, and the stored documents can be retrieved by a StreamServer application connected to the same runtime repository.

In Document Broker Plus “out of the box”, all StreamServer applications must belong to the same application domain. This means the runtime repository in the application domain is used as Post-processing storage.

You can also use custom connection profiles to connect StreamServer applications in different application domains to a common runtime repository. Please contact StreamServe support for assistance.

Adding metadata to documents

Metadata, for example customer number, customer name, etc., can be stored with the documents in the Post-processing storage. This enables applications to use the metadata to search for specific documents in the Post-processing storage.

Post processing documents

Documents retrieved from the Post-processing storage can be post-processed. Post-processing includes sheet layout, sorting, and enveloping of documents. See the *Sheet layout* documentation (*About sheet layout*) and the *Document sorting and bundling* documentation (*About document sorting and bundling*) for more information on post-processing features.

Document Broker for FastObjects

Previous versions of StreamServe Persuasion include a Document Broker solution where documents are stored in a FastObjects database. This solution is still available in StreamServe Persuasion SP5.

You can upgrade Design Center Projects that include Document Broker for FastObjects to Persuasion SP5, and convert the Document Broker solution to Document Broker Plus. See *Converting FastObjects to Document Broker Plus* on page 27. There are however some features in Document Broker for FastObjects (scripting, job context, PPQ editor, etc.) that cannot be used in the Persuasion SP5 version of Document Broker Plus.

Note: You cannot use Document Broker for FastObjects and Document Broker Plus in the same Design Center Project.

Storing documents

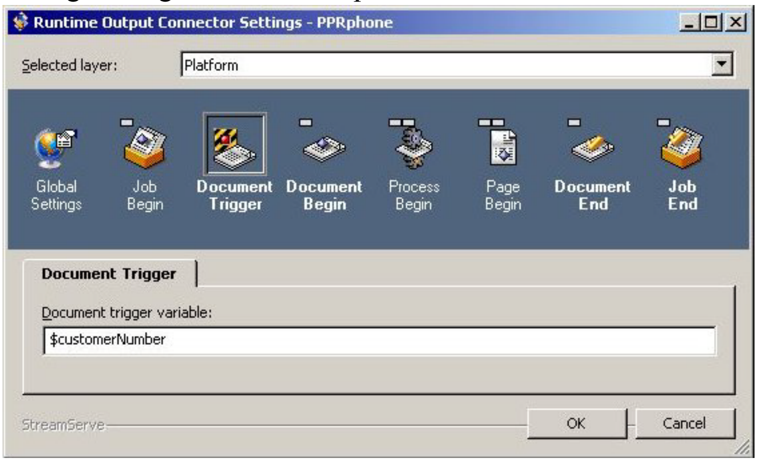
To store documents in a Post-processing storage, you need a specific output connector and driver. You must also specify a document trigger for the output documents, and assign the appropriate document type to the output documents.

Output connector and driver

You must use the `Document Broker Plus` output connector and `SDR for Relational Database` driver. See *Creating a Document Broker Plus output connector* on page 9 for information on how to configure an output connector to store documents in a Post-processing storage.

Document trigger

To store the output as documents, you must create a document trigger that defines what to include in each document. You create the document trigger, e.g. `$customerNumber`, on the Document Trigger tab in the runtime output connector settings dialog box for the output connector.


Runtime Output Connector Settings - PPRphone

Selected layer: Platform

Document Trigger

Document trigger variable: $customerNumber

Figure 2 Using $customerNumber as Document Trigger

Document type resource

You must add the appropriate document type resource to the output connector. The document type defines which Post-processing storage to use, and the metadata to attach to the stored documents. See *Assigning a document type to the output* on page 9 for information on how to assign a document type to an output connector.

Metadata

You can attach metadata to the output documents. Metadata is used by the document retrieving StreamServer application to:

- Search and retrieve stored documents.

- Sort, bundle, and envelope documents.

Metadata is defined in the document type resource added to the output connector. See *Configuring metadata* on page 10 for information on how to configure metadata in a document type.

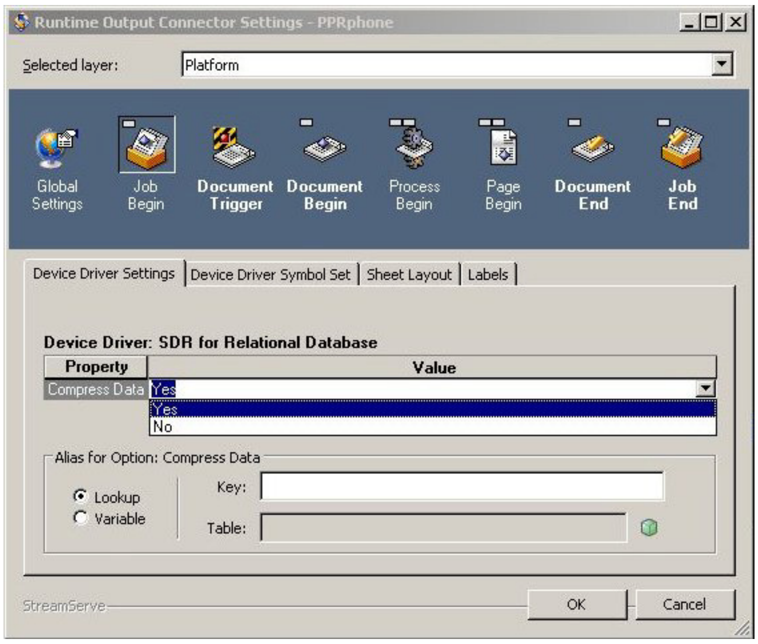
Stored variables

The document retrieving StreamServer application cannot use variables defined in the document storing StreamServer application. To make a variable available after a document is stored, you must store the variable with the document. See *Storing variables with the documents* on page 11 for information on how to store variables.

Compressing the output

Documents are by default compressed before they are stored. This reduces the size of the Post-processing storage and enables documents to be retrieved faster. For smaller jobs it can be more efficient not to compress the data.

You specify whether to compress data on the Device Driver Settings tab at Job Begin in the runtime output connector settings dialog box for the output connector.



Device Driver: SDR for Relational Database

Property	Value
Compress Data	Yes

Yes
No

Figure 3 Compressing output

PPQ for all documents in a stored job

StreamServers that retrieve documents from a Post-processing storage use Post Processor Queries (PPQs) to determine which documents to retrieve.

If you want a StreamServer to retrieve all documents in a job stored in the runtime repository, you can let the StreamServer that stores the jobs create the PPQ for you. See *Auto-generating PPQs* on page 12 for information.

Procedures

In this section

Creating a Document Broker Plus output connector

To store documents in a Post-processing storage, you must use a `Document Broker Plus` output connector and an `SDR for Relational Database` driver.

To create a Document Broker Plus output connector

1. Right-click the Platform view in Design Center and select **New Output Connector** > **Document Broker Plus**. A new output connector is added to the Platform view.
2. Rename the output connector.
3. Double-click the new output connector. The Output Connector Settings dialog box opens.
4. Activate the physical layer and click the **Connector** icon.
5. From the **Connector type** drop-down list, select **Document Broker Plus**.
6. Activate the generic layer and click the **Driver** icon.
7. From the **Device** drop-down list, select **SDR for Relational Database** and click **OK**.

Assigning a document type to the output

You must add the appropriate document type resource to the output connector (runtime output connector settings in Design Center). The document type defines which Post-processing storage to use, and the metadata to attach to the stored documents.

Prerequisites

The document type must be available in a resource set connected to the runtime configuration. See *Creating document types and metadata groups* in the *Document types and metadata* documentation for more information on how to manage document types.

To assign a document type to an output connector

1. In the Runtime configuration view in Design Center, right-click the Process connected to the Document Broker Plus connector and select **Connector Settings**. The Runtime Output Connector Settings dialog box opens.
2. Activate the generic layer and click the **Document End** icon.
3. On the **Document Broker Plus** tab, browse to and select the appropriate document type resource.

Configuring metadata

The document retrieving StreamServer application uses metadata to:

- Search and retrieve documents stored in Post-processing storage.
- Sort and bundle documents.

Metadata is defined in the document type assigned to the output connector that stores the documents.

This means all metadata you want attach to a document must be added as metadata to the corresponding document type (maximum 500 metadata per document type). The metadata added to the document type must also be enabled in Post-processing context. See *Creating document types and metadata groups* in the *Document types and metadata* documentation for more information on how to manage document types.

Storing variables with the documents

Stored variables can be used for scripting and statistics after the documents are retrieved from the Post-processing storage. To make a variable available after a document is stored, you must store the variable with the document at either Document Begin, Process Begin, Page Begin, or Document End.

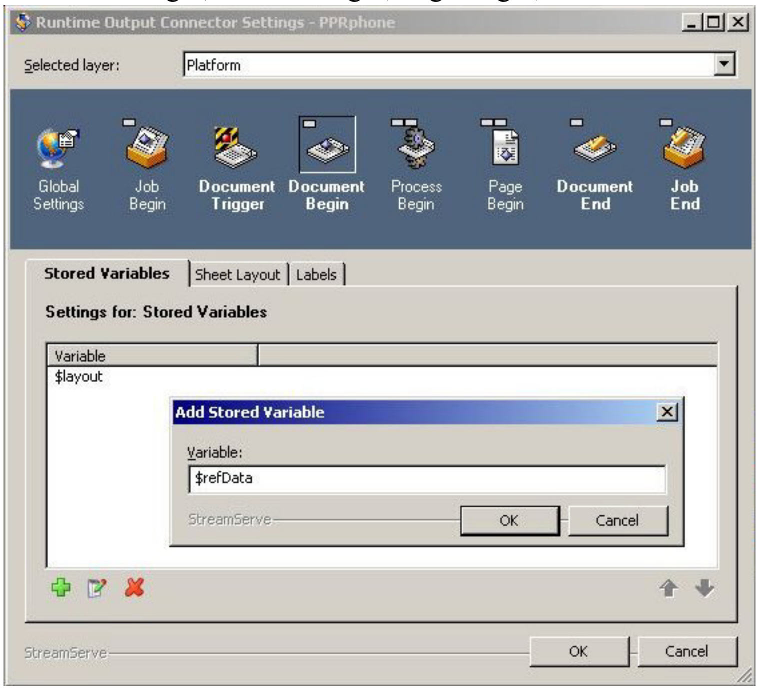


Figure 4 Storing variables at Document Begin

A stored variable is only valid at the same level as it was stored. For example, a variable stored at Document Begin is valid until Document End, whereas a variable stored at Page Begin only is valid until the next Page Begin.

To store a variable

1. In the Runtime configuration view in Design Center, right-click the Process connected to the Document Broker Plus connector and select **Connector Settings**. The Runtime Output Connector Settings dialog box opens.
2. Activate the generic layer and click the **Document Begin**, **Process Begin**, **Page Begin**, or **Document End** icon.
3. Select the **Stored Variables** tab and click ✚. The Add Stored Variable dialog box opens.
4. Enter the variable and click **OK**. The new variable is added to the Stored Variables list.

Auto-generating PPQs

When a StreamServer application stores documents in a Post-processing storage, it can generate a PPQ that can be used by document retrieving StreamServer applications. A PPQ generated this way instructs the retrieving StreamServer application to select and process all documents in a stored batch.

The example below shows a PPQ with instructions to process all documents in the stored batch with ID=8FA23889-7BE7-B743-8842-2607A980C851.

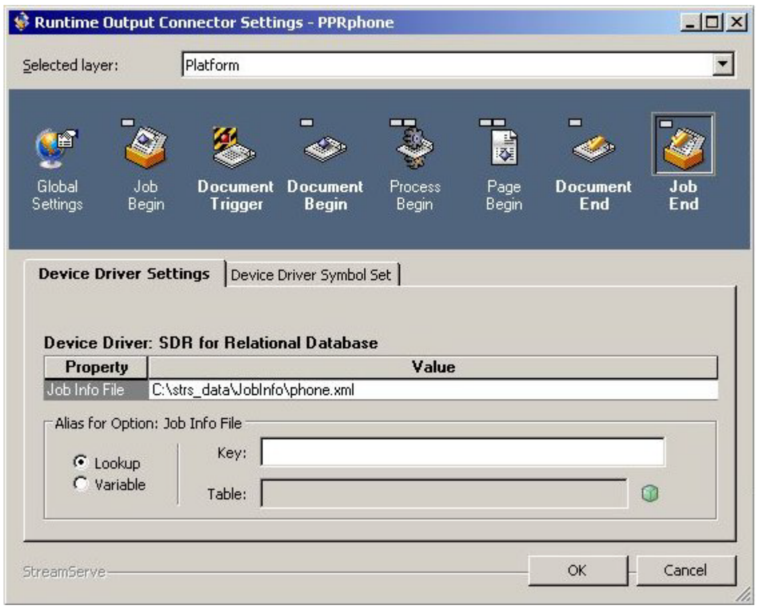
Example 1 *Auto generated PPQ.*

Process all documents in the stored batch with ID=8FA23889-7BE7-B743-8842-2607A980C851.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="process">
  <database>
    <jobset sel="ID='8FA23889-7BE7-B743-8842-2607A980C851'"/>
  </database>
</s-dbs>
```

To auto generate a PPQ

1. In the Runtime configuration view in Design Center, right-click the Process connected to the Document Broker Plus connector and select **Connector Settings**. The Runtime Output Connector Settings dialog box opens.
2. Activate the generic layer and click the **Job End** icon.
3. In the **Job Info File** field, enter the path where to create the PPQ, for example:
   `C:\strs_data\JobInfo\phone.xml`


Runtime Output Connector Settings - PPRphone

Selected layer: Platform

Device Driver: SDR for Relational Database

Property	Value
Job Info File	C:\strs_data\JobInfo\phone.xml

Alias for Option: Job Info File

[✓ Lookup] Variable

Key:

Table:

When the job is finished, and all documents are stored in the repository, the PPQ is created in the path specified. Note that all folders in the path must exist.

Retrieving documents

To retrieve documents from a Post-processing storage, you need a Post-processor, a Post Processor Query (PPQ), and an input connector that collects the PPQ and delivers it to the Post-processor.

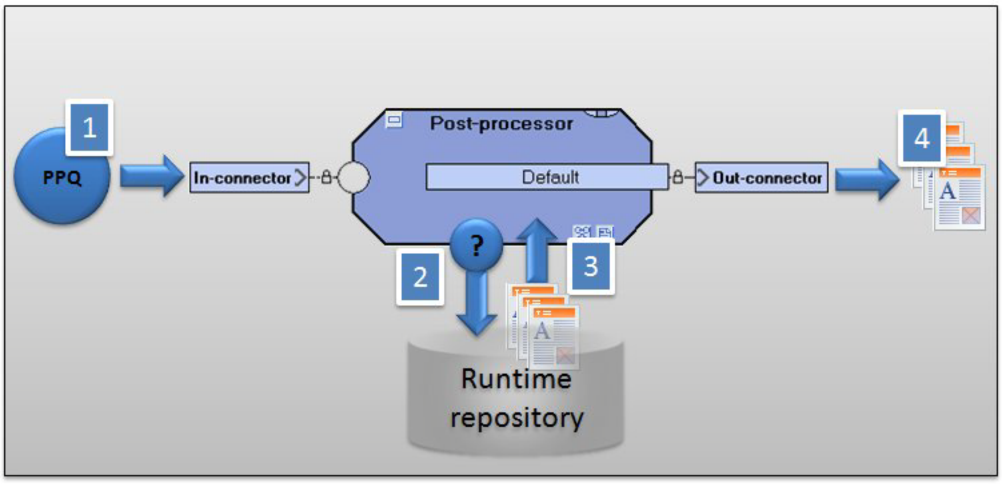


Figure 5 Retrieving documents from a runtime repository

StreamServer retrieves and processes documents as follows (see numbers in figure above):

1 The Post-processor collects a PPQ via an input connector.

2 The Post-processor uses the PPQ to query the runtime repository for documents.

3 The Post-processor retrieves the documents from the runtime repository, and StreamServer post-process the retrieved documents.

4 StreamServer delivers the documents via the appropriate output connector.

Post-processor

A Post-processor is equivalent to a Job in a Runtime configuration. The Post-processor does not include any Message configurations as the Job does, since it retrieves already created documents from a runtime repository.

The Post-processor is connected to input and output connectors in the same way as a runtime Job. The input connector collects the PPQ that specifies which documents to retrieve from the runtime repository, and the Post-processor uses the PPQ to query the runtime repository for documents. When the documents are retrieved, StreamServer can post-process the documents (sheet layout, sorting, enveloping, etc.) and deliver the output via the appropriate output connector.

PPQ

A PPQ (Post Processor Query) is used by the Post-processor to determine which documents to retrieve from a runtime repository. The PPQ is an XML file where different types of elements and attributes specify which documents to retrieve.

You can create a PPQ as a resource in a resource set. This type of PPQ is used with a Post-processor scheduler input connector. You can also create PPQs using any application that can generate XML output, for example an XMLOUT Process that creates a PPQ file in a specific target directory.

Input connector

A Post-processor uses an input connector to collect the PPQ (not the documents from the runtime repository). You can use different types of input connectors to collect PPQs. For example, a Directory input connector that collects the PPQ from a specific directory.

Output connectors

Which type of output connector to use depends on the type of output you want to create. For example, you can use a File output connector to create AFP output.

Post processing

StreamServer can post-process the documents retrieved from the runtime repository. Post-processing includes sheet layout, sorting, and enveloping of documents. See the *Sheet layout* documentation (*About sheet layout*) and the *Document sorting and bundling* documentation (*About document sorting and bundling*) for more information about post-processing features.

Configuring Post-processors

A Post-processor is equivalent to a Job in a Runtime configuration. The Post-processor does not include any Message configurations as the Job does, since it retrieves already created documents from a runtime repository.

The Post-processor is connected to input and output connectors the in same way as a runtime Job. The input connector collects the PPQ that specifies which documents to retrieve from the runtime repository, and the Post-processor uses the PPQ to query the runtime repository for documents. When the documents are retrieved, StreamServer can post-process the documents (sheet layout, sorting, enveloping, etc.) and deliver the output via the appropriate output connector.

Process links

A Post-processor always includes a default Process link, which is used as the connection point to the output connector. To enable Process specific or Page specific output connector settings, you must add a new Process link to the Post-processor job. This Process link must be linked to the Process that stored the corresponding document in the runtime repository.

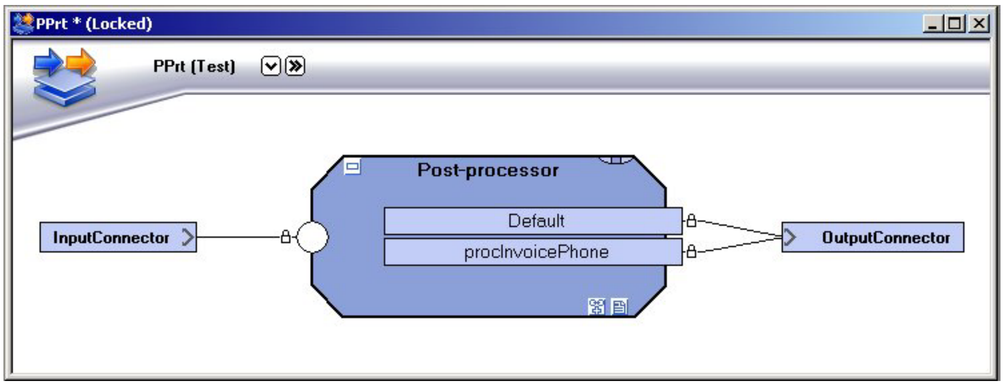


Figure 6 Post-processor with default and specific process links.

To create a Post-processor

1. Right-click the Runtime configuration view and select **New Post-processor**. A new Post-processor is added.
2. Rename the Post-processor.
3. Connect the appropriate input and output connectors to the Post-processor.

To add a new Process link

1. Right-click the Post-processor and select **Add Process Link**.
2. Browse to and select the appropriate Message and Process (you can multi-select Processes). The new Process links are added to the Post-processor, and connected to the same output connector as the Default Process link.

Activating Document Broker Plus

When you create a Post-processor, it will by default try to connect to a FastObjects database. You must, in each runtime configuration containing Post-processors, activate Document Broker Plus.

Note: You cannot revert to the FastObjects solution once you have activated Document Broker Plus.

To activate Document Broker Plus

Right-click the Runtime configuration view and select **Document Broker Plus**.

Creating new PPQs

You can create PPQs using any editor or application that generates XML output.

In this section

PPQ syntax

The PPQ syntax is illustrated below.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="value">
 <database>
  <docset sel="filter" metainfo="filter"/>
  ...
 </database>
</s-dbs>
```

- The `<s-dbs>` element specifies which action to apply to the selected documents. See *Action* on page 19.
- The `<docset>` element specifies which documents in the repository to select. See *Document selection* on page 20.

Action

Use the `action` attribute in the `<s-dbs>` element to specify what to do with the selected documents.

Attribute value	Description
`process`	Process the documents, but do not delete them in the repository.
`process_and_delete`	Process the documents, and delete them in the repository.
`delete`	Delete the documents in the repository.

Example 2 *Action=process.*

Process all documents.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="process">
    <database>
          <docset sel="all"/>
    </database>
</s-dbs>
```

Example 3 *Action=process_and_delete.*

Process all documents, and delete them in the repository.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="process_and_delete">
    <database>
          <docset sel="all"/>
    </database>
</s-dbs>
```

Document selection

The `<docset sel="filter" metainfo="filter"/>` sections in the PPQ specify which documents in the runtime repository to select. You can use two types of filters:

- Document property filters (`sel="filter"`). See *Creating document property filters* on page 21.
- Metadata filters (`metainfo="filter"`). See *Creating metadata filters* on page 23.

Filter operators

You can use the operators shown in the table below when you create document property filters and metadata filters.

Operator	**Description**
=	Equal to. Numeric and string values. For example: `metainfo="CustomerName=&quot;Eva Farrel&quot;"`

Operator	**Description**
`< (&lt;)`	Less than. Numeric and string values. For example: `metainfo="CustomerNumber&lt;1020"`
`> (&gt;)`	Greater than. Numeric and string values.For example: `metainfo="CustomerNumber&gt;1020"`
`!=`	Not equal to. Numeric and string values. For example: `metainfo="CustomerNumber!=1020"`
`>= (&gt;=)`	Greater than or equal to. Numeric and string values. For example: `metainfo="CustomerNumber&gt;=1020"`
`<= (&lt;=)`	Less than or equal to. Numeric and string values. For example: `metainfo="CustomerNumber&lt;=1020"`
`+`	Addition. Numeric and string values.
`-`	Subtraction. Numeric values only.
`*`	Multiplication. Numeric values only.
`/`	Division. Numeric values only.
`AND`	Logical AND. For example: `metainfo="CustomerNumber&lt;=1020 AND Country=SWE"`
`OR`	Logical OR. For example: `metainfo="Country=FIN OR Country=SWE"`

Creating document property filters

You can use the `sel` attribute in the `<docset>` element to create a filter that uses document properties as filter criteria.

Document properties		
Property	**Description**	**Examples**
`All`	Select all documents in the repository.	`sel="All"`
`ID`	Select a document with a specific DocumentAbstractionID or DocSequenceNumber.	(DocumentAbstractionID) `sel="ID='8FA23889-7BE7-B743-8842-2607A980C851'"` (DocSequenceNumber) `sel="ID=2"`
`Priority`	Select documents with a specific priority.	`sel="Priority=50"`

Document properties		
Property	**Description**	**Examples**

`CreationTime`	Select documents created a specific date and time.	`sel="CreationTime=2010-03-29"`
		`sel="CreationTime&gt;2010-03-29T09:30:00"`

`ProcessTime`	Select documents updated a specific date and time.	`sel="ProcessTime=2010-03-29"`
		`sel="ProcessTime&gt;2010-03-29T09:30:00"`

`Error`	Select documents with a specific Error ID.	`sel="Error=1"`
`Status`	Select documents of a specific Status.	`sel="Status=Stored"`

`State`	Select documents of a specific State.	`sel="State=Submitted"`
		`sel="State=1"`

`DocumentType`	Select documents connected to a specific document type. The document type can be specified using the name or GUID of the document type.	`sel="DocumentType='Invoice'"`
		`sel="DocumentType='98A23889-62E7-B718-8842-2607A980C851'"`

`PageCount`	Select documents that contain a specific number of pages.	`sel="PageCount&gt;=10"`

Example 4 *sel=all.*

Document property filter selecting all documents in the repository.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="process">
    <database>
          <docset sel="all"/>
    </database>
</s-dbs>
```

Date values

Date values must follow the ISO 8601 standard. Local time with offset to UTC (Coordinated Universal Time) is not supported. Examples of ISO 8601 date values are shown in the table below.

Calendar date	`YYYY-MM-DD` For example: `2010-05-20`

Date and time	`YYYY-MM-DDThh:mm:ss` For example: `2010-05-20T13:04:25`

Creating metadata filters

You can use the `metainfo` attribute in the `<docset>` element to create a filter that uses metadata stored with the documents as filter criteria.

Example 5 *metainfo="Country=SWE"*

Metadata filter selecting documents where metadata Country=SWE.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="process">
    <database>
          <docset metainfo="Country=SWE"/>
    </database>
</s-dbs>
```

The metadata must be included in the document type resource connected to the output connector, and the metadata must also be enabled in Post Process context.

Using a Job Info File as PPQ

When you store documents in a runtime repository, you can let StreamServer create a Job Info File when all documents in the processed job are successfully stored in the runtime repository (see *Auto-generating PPQs* on page 12). The Job Info File is stored in a directory specified on the output connector that delivered the output to the runtime repository.

Example 6 *Using a Job Info File PPQ.*

Job Info File.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="process">
    <database>
        <jobset sel="ID='8FA23889-7BE7-B743-8842-2607A980C851'"/>
    </database>
</s-dbs>
```

You can use the Job Info File as a PPQ. If you use a Directory input connector that scans the Job Info File directory, the PPQ is collected when all documents are stored in the runtime repository.

Using a Post-processor scheduler input connector

You can use different types of standard input connectors to collect PPQs, for example Directory and HTTP input connectors. You can also use a Post-processor scheduler input connector. The Post-processor scheduler input connector is actually intended for Document Broker for FastObjects, but it can also be used for Document Broker Plus.

PPQ resource

The Post-processor scheduler input connector is used together with a PPQ resource. In a Document Broker for FastObjects scenario you should use a Post-processor Query resource. This resource type uses an editor that must connect to a FastObjects repository, which means you cannot use it in a Document Broker Plus scenario. In a Document Broker Plus scenario you must use a text editor resource type, for example a Sample resource, where you can type in the appropriate PPQ xml.

To configure a Post-processor scheduler input connector

1. Create a PPQ resource (e.g. a Sample resource) where you enter the appropriate PPQ xml.
2. Create a Post-processor scheduler input connector.
3. Open the Input Connector settings dialog box (physical layer) for the connector.
4. Browse to and select the **Post-processor query** you created in Step 1.
5. Use the **Schedule** property to schedule when and how often to collect the PPQ (e.g. once a month).
6. Click **OK**.

How the Post-processor scheduler input connector works

The Post-processor scheduler input connector is similar to a Directory input connector. It periodically scans a resource directory in the StreamServer application’s working directory for input. For example, if you use a Sample resource as PPQ, the connector scans the Sample resource directory for input.

When you export and deploy a Project, the PPQ is added to the resource directory. The Post-processor scheduler input connector collects, but does not delete, the PPQ as soon as the StreamServer application is started. The Post-processor scheduler input connector then waits until the next collection event (specified by the **Schedule** property), collects the same PPQ once again, waits until the next collection event, collects the PPQ and so on.

Converting FastObjects to Document Broker Plus

Design Center Projects that include Document Broker for FastObjects can be converted to use Document Broker Plus in StreamServe Persuasion SP5.

Documents stored in the FastObjects database cannot be migrated to the Post-processing storage in the runtime repository, which means all Document Broker for FastObjects jobs must be finished before you change to Document Broker Plus.

Note: You cannot use Document Broker for FastObjects and Document Broker Plus in the same Design Center Project.

In this chapter

Converting document store configurations

Output connector and driver

In Document Broker for FastObjects you used a `Post-processor repository (Legacy)` output connector and an `SDR` driver to store documents in the FastObjects repository. To use Document Broker Plus you must convert all FastObjects output connectors and drivers to Document Broker Plus. See *Converting to Document Broker Plus output connectors* on page 27.

Metadata and Post-processing storage

In Document Broker for FastObjects you used the Runtime Output Connector Settings dialog box (Metadata Keys tab) to specify which metadata to attach to the documents stored via the output connector. To use Document Broker Plus you must specify the metadata in a document type resource that you connect to the output connector. See *Connecting metadata to Document Broker Plus output connectors* on page 28.

Converting to Document Broker Plus output connectors

You must convert all `Post-processor repository (Legacy)` connectors in your Project to `Document Broker Plus` output connectors.

To convert to Document Broker Plus connectors

1. Activate the Platform view and select **Platform** > **Convert to Document Broker Plus**. A confirmation dialog opens. The path to the log file is shown in this dialog. Make a note of this path.
2. Click **Yes**. You are prompted to open the log file.
3. Click **Yes**. The connectors are converted to Document Broker Plus, and the log file opens.

The log file shows the result of the conversion. In the log you can see that all metadata defined before conversion to Document Broker Plus are dropped. For each converted output connector you can see all dropped metadata (`$<metadata>`) and the type of metadata (`Num` or `Str`).

You must re-connect the dropped metadata to the converted output connectors. See *Connecting metadata to Document Broker Plus output connectors* on page 28.

Connecting metadata to Document Broker Plus output connectors

In the `Post-processor repository (Legacy)` connectors, metadata was defined at Document End in the Runtime Output Connector settings. After conversion to Document Broker Plus, all defined metadata is dropped from the output connectors .

To re-connect the dropped metadata to a Document Broker Plus output connector you must create a new document type resource (or use an existing document type resource) and connect it to the Document Broker Plus output connector. The document type resource that you connect must include all metadata that was dropped from the old output connector.

To create a document type resource

See *Creating document types and metadata groups* in the *Document types and metadata* documentation.

To connect a document type resource to an output connector

See *Assigning a document type to the output* on page 9.

Converting document retrieve configurations

Activating Document Broker Plus

You must, in each runtime configuration containing Post-processors, activate Document Broker Plus.

See *Activating Document Broker Plus* on page 18.

Reusing Post Processor Queries (PPQs)

PPQs used in Document Broker for Fast Objects can, under certain circumstances, be used in Document Broker Plus. See *Reusing PPQs* on page 29.

Unsupported script functions

There are a number of script functions the can be used in Document Broker for FastObjects, but cannot be used in Document Broker Plus. See *Unsupported script functions* on page 29.

Reusing PPQs

You can, under certain circumstances, reuse old PPQs in Document Broker Plus.

Post-processor Query editor

The Post-processor Query resource editor cannot be used to edit PPQs for Document Broker Plus. This resource editor can only be used to edit FastObjects queries.

The <jobset> element

The `<jobset>` element in PPQs can only be used to select “all” documents (`<jobset sel="all">`) or documents related to a specific “stored job ID” (`<jobset sel="`*`ID`*`">`).

Unsupported script functions

The following post-processor script functions cannot be used in Document Broker Plus:

- `CurrJobProperty`
- `GetJobProperty`
- `GetSegJobProperty`
- `GetPpJobProperty`
- `GetPpReposProperty`
- `GetReposProperty`
- `GetSegReposProperty`

- `UpdatePPDocStatus`