OPENTEXT

OpenText StreamServe 5.6.1

Document Broker Plus

User Guide

Rev A

OpenText StreamServe 5.6.1 Document Broker Plus User Guide
Rev A

Open Text SA
40 Avenue Monterey , Luxembourg, Luxembourg L-2163
Tel: 35 2 264566 1

Open Text Corporation
275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1
Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Email: support@opentext.com
FTP: ftp://ftp.opentext.com
For more information, visit http://www.opentext.com

Copyright ©2013 Open Text Corporation

OpenText is a trademark or registered trademark of Open Text SA and/or Open Text ULC. The list of trademarks is not exhaustive of other trademarks, registered trademarks, product names, company names, brands and service names mentioned herein are property of Open Text SA or other respective owners.

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Contents

About Document Broker Plus

Document Broker Plus enables consolidation of documents generated by several StreamServer applications by storing the documents in a common Document Broker repository. The documents can be collected by any StreamSever application that has access to the same Document Broker repository.

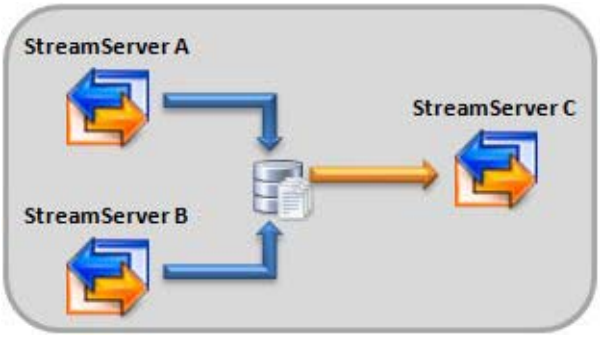


Figure 1 Document Broker Plus scenario with two storing StreamServer applications and one collecting StreamServer application

Storing documents in a Document Broker repository

To store documents in a Document Broker repository, you need a Document Broker Plus output connector and driver. You must also specify a document trigger for the documents.

Adding metadata to stored documents

To be able to search for documents in the Document Broker repository you must add metadata to the documents. You do this you must first create a document type resource that includes all metadata to use, and then add this document type resource to the Document Broker Plus output connector.

Collecting documents from a Document Broker repository

To collect documents from a Document Broker repository you need an input connector, a Post-processor and a query. The query includes document type and metadata filters that define which documents to collect from the Document Broker repository. This query is delivered to the Post-processor via the input connector, and the Post-processor uses the information in the query to collect the documents.

Post processing documents

Documents collected from a Document Broker repository 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 include a Document Broker solution where documents are stored in a FastObjects database. This solution is still available in this version of StreamServe.

You can upgrade Design Center Projects that include Document Broker for FastObjects to Document Broker Plus. See *Upgrading FastObjects Projects to Document Broker Plus* on page 12. There are however some features in Document Broker for FastObjects (scripting, job context, PPQ editor, etc.) that cannot be used in Document Broker Plus.

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

Concepts

In this section

Document Broker repository

The Document Broker repository is where the documents are stored. By default, the Document Broker repository resides in the runtime repository of an application domain. This means all StreamServer applications in an application domain use the runtime repository as Document Broker repository.

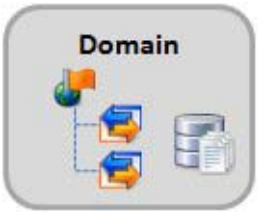


Figure 2 Document Broker repository in runtime repository

This default solution can only be used if you know that the Document Broker repository will never be shared by StreamServer applications in different application domains at anytime in the future. It is however not recommended to use the runtime repository as a Document Broker repository, as it limits the solution to only being able to consolidate documents created within a particular application domain. The recommendation is therefore to create a central Document Broker repository that you can link to one or more application domains.

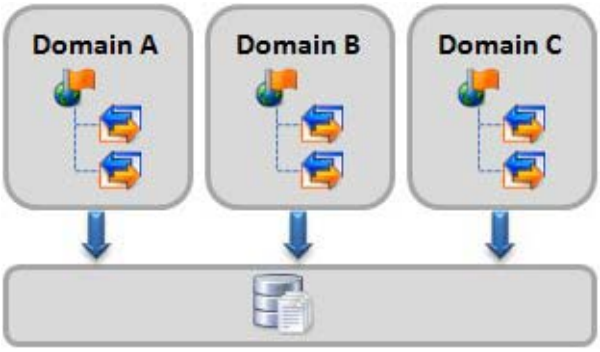


Figure 3 Multiple application domains linked to a central Document Broker repository

Post-processor

A Post-processor is equivalent to a Job in a Runtime configuration. The Post-processor retrieves documents from a Document Broker repository, and does not include any Message configurations as the Job does.

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

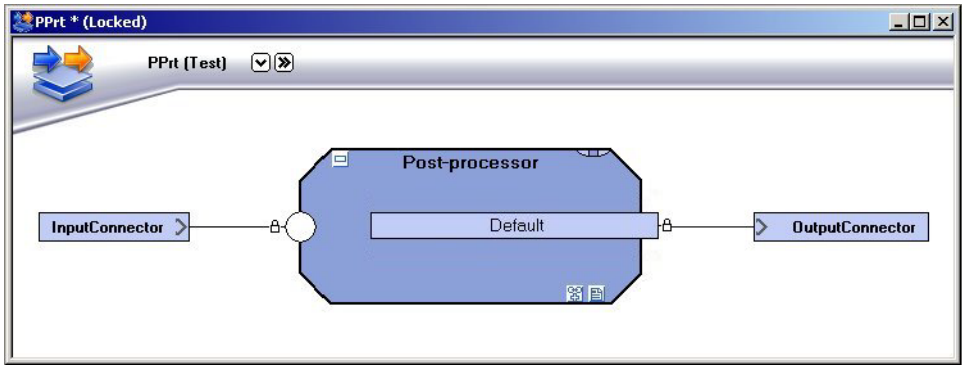


Figure 4 Post-processor

Queries

Queries contain filters based on document types and metadata, and are used by Post-processors to query the Document Broker repository for documents. A query specifies which documents to select, and the action to apply to the selected documents:

- Process the selected documents.
- Process and delete the selected documents.
- Delete the selected documents.

Document Broker Plus supports two different query syntaxes:

- **Document Broker Query (DBQ)** – syntax developed for Document Broker Plus. You can use Document Broker Query Tool to create DBQs.
- **Post-Processor Query (PPQ)** – syntax developed for Document Broker for FastObjects. In Document Broker Plus you can use manually created or auto-generated PPQs.

Related topics

- *Creating PPQs* on page 41

Preparing the Document Broker Plus Environment

In this section

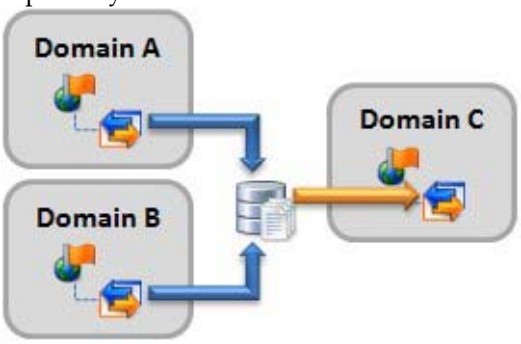
Creating a central Document Broker repository

It is recommended to create a central Document Broker repository that can be linked to several application domains. By using a central Document Broker repository you will have one single Document Broker repository used and managed for all document broking and post processing requirements. This central repository enables you to consolidate documents from multiple sources in different application domains.

Note: You cannot connect an application domain to several Document Broker repositories.

Scenario

Company ABC uses three different StreamServer applications, all running in separate application domains. Company ABC needs to consolidate the output documents from StreamServer A and StreamServer B, and use StreamServer C to deliver the consolidated documents to the customers. To achieve this, a central Document Broker repository is created and connected to all three application domains. StreamServer A and StreamServer B stores the output documents in this central repository, and StreamServer C collects the documents from the same repository.



To create a central Document Broker repository

1. In Control Center, right-click the **Repositories** node and select **New Repository**. The Repository dialog box opens.
2. From the **Type** drop-down list, select **Document Broker Repository**.
3. Enter the **Name** and a **Description** of the Document Broker repository and click **OK**. The Configuration dialog box opens,
4. Configure the database settings for the repository and the connection settings to access the repository.
5. Right-click the new Document Broker repository node and select **Create Database**. The Create Database dialog box opens.
6. Select the appropriate Operation (**Create now** etc.) and click **Start**.

To link a Document Broker repository to application domains

1. Right-click the Document Broker repository node and select **Link Application Domain**. The Link Application Domain dialog box opens.
2. Select the application domains to link the Document Broker repository to, and click **OK**.

Activating Document Broker Plus in Design Center

When you create a Post-processor in Design Center, 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 return to FastObjects once you have activated Document Broker Plus.

To activate Document Broker Plus

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

Upgrading FastObjects Projects to Document Broker Plus

Design Center Projects that include Document Broker for FastObjects can be upgraded to Document Broker Plus.

Documents stored in the FastObjects database cannot be migrated to the Document Broker repository, which means all Document Broker for FastObjects jobs must be finished before you upgrade 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

Upgrading document storage configurations

Design Center Projects configured to store documents in a FastObjects database can be upgraded to store documents in a Document Broker repository.

Output connector and driver

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 12.

Metadata

In Document Broker for FastObjects, metadata was specified as runtime output connector settings. To use Document Broker Plus you must specify the metadata in a document type resource. See *Connecting metadata to Document Broker Plus output connectors* on page 13.

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 13.

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 document type resource that you connect to the Document Broker Plus output connector.

Creating a document type resource

You must create a document type resource that includes all metadata dropped from the old output connector. The metadata must be enabled in Post-processing context. See *Creating document types and metadata groups* in the *Document types and metadata* documentation for information about how to create document types.

To connect the document type resource to the 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.

Upgrading document collection configurations

Design Center Projects configured to collect documents from a FastObjects database can be upgraded to collect documents from a Document Broker repository.

Activating Document Broker Plus

You must activate Document Broker Plus in all runtime configurations that contain Post-processors. See *Activating Document Broker Plus in Design Center* on page 11.

Reusing Post Processor Queries (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" and/or document type (`<jobset sel="ID = &quot;some_id_here&quot; and DocumentType = &quot;some_document_type_id_here&quot;">`)

Unsupported script functions

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

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

Storing documents in a Document Broker repository

You need the following Design Center components to store documents in a Document Broker repository:

- Document Broker Plus output connector
- Document type resource containing metadata

Overview

StreamServer processes and stores documents as follows:

1. Input is received via an input connector.
2. StreamServer processes the input data.
3. Metadata is added to the documents.
4. StreamServer stores the documents in the Document Broker repository.

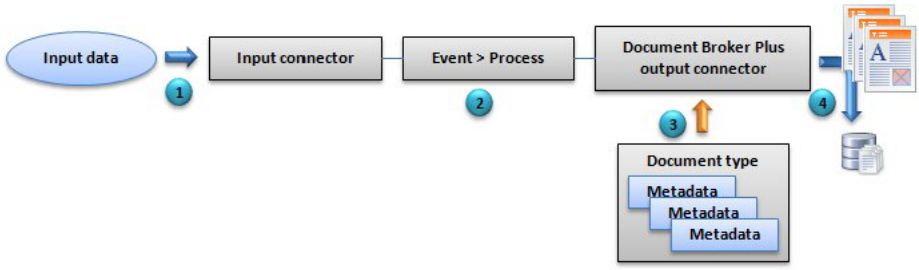


Figure 5 Storing documents in a Document Broker repository

Document Plus output connector

You must use the **Document Broker Plus** output connector to store documents in a Document Broker repository. You must select the **SDR for Relational Database** driver and create a document trigger in order to store the output as documents in the Document Broker repository.

Compression of the documents (enabled by default) reduces the size of the Document Broker repository. For smaller jobs it can be more efficient not to compress documents. You specify whether to enable compression in the runtime output connector settings (Device Driver Settings tab at Job Begin).

Document type resource

You must create a document type resource and connect it to the Document Broker Plus output connector. The document type must include all metadata to add to the stored documents. The metadata is used as search criteria in the Document Broker repository, and to sort and bundle documents collected from the Document Broker repository.

Configuring metadata

When you store documents in the Document Broker repository you must add metadata to the documents. This metadata is used as search criteria in the Document Broker repository, and to sort and bundle documents collected from the Document Broker repository.

To be able to add metadata to the documents you must first create a document type resource for each document type (Invoice, Order, etc.) to store in the Document Broker repository. The document type resource must contain all metadata to add to the documents (maximum 500 metadata per document type), and this metadata must 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.

Configuring a Document Broker Plus output connector

To be able to store documents in a Document Broker repository you must first create a Document Broker Plus output connector with an SDR for Relational Database driver. Then you connect the Process to the Document Broker Plus output connector, specify a document trigger for the documents, and add a document type resource to the connector. You can also configure the output connector to auto-generate a PPQ, as a file and/or as a call to another connector.

Prerequisites

The document type resource (see *Configuring metadata* on page 16) must be available in a resource set connected to the runtime configuration.

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. Double-click the new output connector. The Output Connector Settings dialog box opens.
3. Activate the generic layer and click the **Driver** icon.
4. From the **Device** drop-down list, select **SDR for Relational Database** and click **OK**.

To create a document trigger

1. In the Runtime configuration view in Design Center, open the Runtime Output Connector Settings dialog box for the Document Broker Plus connector.
2. Activate the generic layer and click the **Document Trigger** icon.
3. Enter the **Document trigger variable**.

To add a document type resource to a Document Broker Plus output connector

1. In the Runtime Output Connector Settings dialog box for the Document Broker Plus connector, activate the generic layer and click the **Document End** icon.
2. On the **Document Broker Plus** tab, browse to and select the document type resource and click **OK**.

To configure the connector to generate PPQ

1. In the Runtime Output Connector Settings dialog box for the Document Broker Plus connector, click the **Job End** icon.
2. On the **Device Driver Settings** tab, enter a PPQ file name and/or the name of the connector. For information about the settings on the tab, see *Connector settings for auto-generated PPQs* on page 51.

Related topics

- *Storing variables with the documents* on page 18
- *Auto-generating PPQs* on page 50

Storing variables with the documents

Stored variables can be used for scripting and statistics after the documents are retrieved from the Document Broker repository. 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.

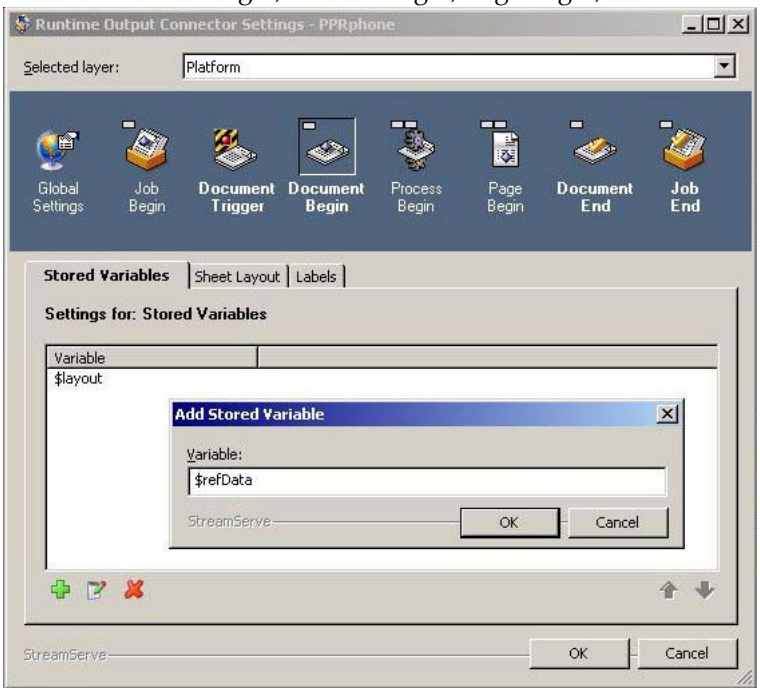


Figure 6 Storing variables at Document Begin

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, open the Runtime Output Connector Settings dialog box for the Document Broker Plus connector.
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.

Storing Document Broker Plus blobs on a disk

By default, Document Broker Plus blobs are stored in the Document Broker repository. In scenarios where you run Document Broker Plus with a few StreamServer applications on a single computer, you can improve performance by storing the Document Broker Plus blobs as files on a disk. For more information, see *Advanced tab* in *Design Center* documentation.

Deleting Document Broker Plus blob files from a disk

To delete the blob files from the disk, you must run one of the following DBQ or PPQ queries:

- delete
- process and delete

Storing successful documents when job fails

By default, documents are stored in the Document Broker repository only if the entire job is processed successfully. If any of the documents in the job fails, no documents are stored in the Document Broker repository. This default behavior prevents StreamServer from storing duplicates when it retries to process job that failed.

You can add the keywords `failjob ignore` to the Document Broker Plus output connector to override this default behavior, and store all successful documents instead of no documents when a job fails.

To add the keywords

1. In Design Center, select **Edit** > **Custom Settings**.
2. In the Edit Custom fields dialog box, browse to and select the **Logical** node of the Document Broker Plus connector.
3. Enter the keywords `failjob ignore` and click **OK**.

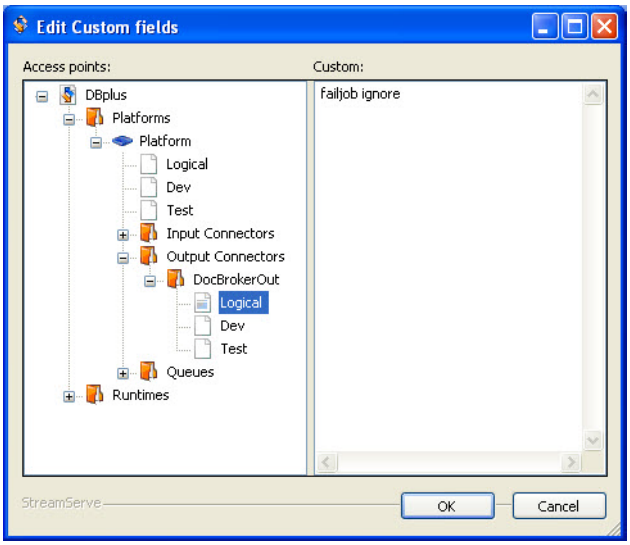


Figure 7 Adding the keywords

Collecting documents from a Document Broker repository

You need the following Design Center components to collect documents from a Document Broker repository:

- Query (DBQ or PPQ)
- Input connector for the query
- Post-processor

Overview

StreamServer collects and processes documents as follows:

1. The Post-processor receives a DBQ or PPQ via an input connector.
2. The Post-processor uses the DBQ/PPQ to query the Document Broker repository for documents.
3. The Post-processor collects the documents from the Document Broker repository, and StreamServer post-processes the collected documents.
4. StreamServer delivers the documents via the appropriate output connector.

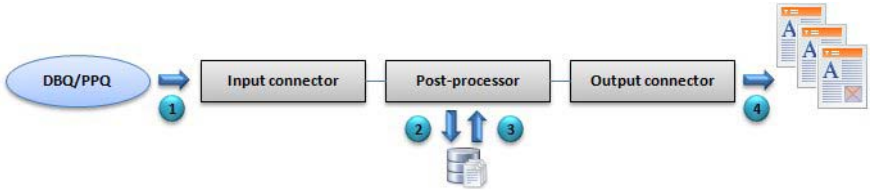


Figure 8 Collecting documents from a Document Broker repository

Query

The query is used by the Post-processor to query the Document Broker repository for documents. The query specifies which documents to select, and the action to apply to the selected documents:

- Process the selected documents.
- Process and delete the selected documents.
- Delete the selected documents.

Input connector

The input connector collects the query. You can use different types of input connectors to collect queries. For example, a Directory input connector that collects the query from a specific directory.

Post-processor

The Post-processor executes the query and collects the documents from the Document Broker repository.

Post-processing

StreamServer can post-process the collected documents. Post-processing includes sheet layout, sorting, and bundling 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.

Creating queries

The query is used by the Post-processor to query the Document Broker repository for documents. The query specifies which documents to select, and the action to apply to the selected documents:

- Process the selected documents.
- Process and delete the selected documents.
- Delete the selected documents.

Document Broker Plus supports two different query syntaxes:

- **Document Broker Query (DBQ)** – syntax developed for Document Broker Plus. You can use Document Broker Query Tool to create DBQs. See *Creating DBQs* on page 31.
- **Post-Processor Query (PPQ)** – syntax developed for Document Broker for FastObjects. In Document Broker Plus you can use manually created or auto-generated PPQs.

Related topics

- *Creating PPQs* on page 41

Configuring input connectors to collect queries

You can use several types of standard input connectors (Directory, HTTP, etc.) to collect queries. Use standard procedures to configure the connector to collect the queries in this case. You can also use the following input connectors:

- Post-processor scheduler (see *Using a Post-processor scheduler input connector* on page 24).
- Service Request (see *Using a Service Request input connector*).

Using a Post-processor scheduler input connector

You can use a Post-processor scheduler input connector to collect the query. The query (DBQ or PPQ) must be included in a resource set connected to the platform.

Creating a query resource

There is no specific resource type for DBQs and PPQs used in Document Broker Plus. You can use any type of text formatted resource, for example a sample resource.

- To create a PPQ resource you must create a new resource, start the resource editor, enter the appropriate PPQ xml and save the resource.
- To create a DBQ resource you can export a DBQ from Document Broker Query Tool, and import the DBQ to the resource set.

To configure a Post-processor scheduler input connector

1. Create a query resource.
2. Create a Post-processor scheduler input connector.
3. Open the Input Connector settings dialog box (physical layer).
4. Browse to and select the **Post-processor query** you created in Step 1.
5. **Schedule** when to collect the query (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 query, the connector scans the Sample resource directory for input.

When you export and deploy a Project, the query is added to the resource directory. The Post-processor scheduler input connector collects, but does not delete, the query 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 query once again, waits until the next collection event, collects the query and so on.

Using a Service Request input connector

You can submit DBQs via a Service Request input connector. For example, if you want to submit a query directly from DBQ Tool to a running StreamServer application, the StreamServer application must retrieve the DBQ via a Service Request input connector.

To configure a Service Request connector

1. Create an input connector.
2. Open the Input Connector Settings dialog box (physical layer).
3. From the **Connector type** drop-down list, select **Service Request**.
4. From the **Request type** drop-down list, select **Generic**.
5. Enter the **Service name** and click **OK**.

Configuring Post-processors

A Post-processor collects documents from a Document Broker repository, and does not include any Message configurations. The Post-processor retrieves a query via an input connector, executes the query, collects the documents from the Document Broker repository and delivers the documents via the appropriate output connector.

Output connector limitations

The output mode for the output connector must not be Process, and the following connectors cannot be used as Post-processor output connectors:

- MAPI for MailOUT
- SMTP (MIME) for MailOUT
- Topcall
- SMS
- Post-processor repository (SQL)
- Post-processor repository (Legacy)

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. This Process link must be linked to the Process that stored the corresponding document in the Document Broker repository.

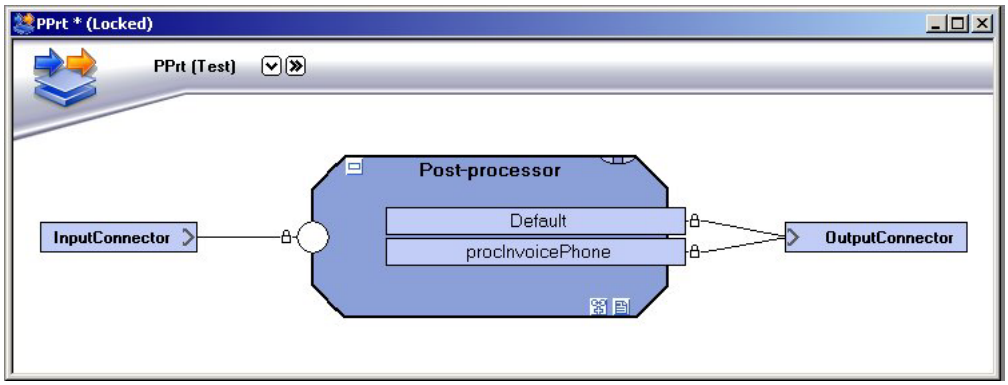


Figure 9 Post-processor with default and specific process links.

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.

To activate Document Broker Plus

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

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

Multiple selection of a static document

You can use a DBQ to select a static document stored in the Document Broker repository more than once. The selection must be based on a unique Document ID.

Purpose

This makes it possible to associate the static document with other transactional documents that are processed and stored in the same Document Broker repository.

Comments

To avoid deleting the static document, you cannot run a process-and-delete DBQ. To delete the transactional documents, you must run a separate delete DBQ.

Scenario

A recall notice is stored in the Document Broker repository. The recall notice contains no transactional content, and the same recall notice is sent for each document set. To delete the documents, exclusive of the recall notice, a second delete query that does not include the Document ID of the recall notice is run.

Script functions for updating metadata

Document Broker Plus post processing scripts support the following script functions to update document metadata in the Document Broker repository:

- *SaveCurrMetadata*
- *SaveSegMetadata*
- *SavePPMetadata*

Using only these script functions for updating metadata in the Document Broker repository is inefficient as they only can store one metadata item at time. This results in heavy database traffic and poor performance for Projects that require metadata update. To solve this problem, you can use the following cache script functions in addition to the `Save<xx>Metadata` functions:

- *PPBeginDocMDUpdate*
- *PPCommitDocMDUpdate*
- *PPRollbackDocMDUpdate*

These cache script functions will cache the metadata updated by `Save<xx>Metadata`, and then store the cached metadata in the Document Broker repository.

You must add `PPBeginDocMDUpdate` before you execute the `Save<xx>Metadata` functions in order to make StreamServer cache the metadata instead of storing the metadata directly in the Document Broker repository. Then you add `PPCommitDocMDUpdate` after the `Save<xx>Metadata` functions are executed in order to send the metadata from the cache to the Document Broker repository.

If you at some point want to stop storing cached metadata in the Document Broker repository, you can use the `PPRollbackDocMDUpdate` function. This function can be called based on some conditions, and when executed it clears all metadata that is still in the cache.

Script execution levels

You can call the cache script functions at any post-processor script level. For example, if `PPBeginDocMDUpdate` is called at Job Begin and `PPCommitDocMDUpdate` is called at Job End, all metadata belonging to all documents are saved when the post-processor job finishes. The consideration here could be memory used by cache for really big jobs.

Note: The scope cannot be nested. A second call to `PPBeginDocMDUpdate` without first calling `PPCommitDocMDUpdate` or `PPRollbackDocMDUpdate` will return an error.

Improving performance in upgraded Projects

If you upgrade a Document Broker for FastObjects Project to Document Broker Plus, and if the Projects contains `Save<xx>Metadata` functions, you can improve performance by just inserting `PPBeginDocMDUpdate` before and `PPCommitDocMDUpdate` after the `Save<xx>Metadata` functions are executed.

Creating DBQs

Document Broker Query (DBQ) Tool is used to create queries to run against Document Broker repositories. DBQs can only be used in a Document Broker Plus environment, and not in a Document Broker for FastObjects environment.

In this section

Getting started with DBQ Tool

You must deploy a DBQ application to your Java application server before you can start DBQ Tool. To be able to use DBQ Tool correctly, you must also run a service gateway application in each application domain included in the same Document Broker Plus environment.

Installation prerequisites

Before you install DBQ Tool, the following must be installed:

- A Java application server.
- Java Runtime Environment on the same computer as the Java application server.

In this section

Recommended settings

Screen resolution

- Recommended screen resolution – At least 1280 x 1024 pixels.
- Minimum screen resolution – 1024 x 768 pixels.
- Run the applications in a maximized browser window.

JVM memory settings

You must make sure there is enough memory available for DBQ Tool on the JVM (Java Virtual Machine).

The amount of memory required depends on the installation environment, the number of deployed web applications, and the load on the Java application server. For example, deploying DBQ Tool to the same Java application server as the StreamStudio web portal requires more memory than deploying to a separate Java application server. We recommend that you use a Java profiling tool for monitoring and tuning the Java memory settings on a running Java application server.

When deploying DBQ Tool to a separate Java application server, we recommend the following memory settings (case sensitive):

JVM	Recommended settings
32-bit JVM	`-Xmx768m` `-XX:MaxPermSize=256m`

64-bit JVM	`-Xmx1024m`
	`-XX:MaxPermSize=384m`
	Note: If you run DBQ Tool on a 64-bit system with 4 GB memory or less, we recommend that you use a 32-bit JVM. A 64-bit JVM is not recommended unless you have more memory – you might need twice as much physical memory on a 64-bit system as on a 32-bit system to achieve the same performance.

For information on how to specify the memory settings, see the user documentation for the Java application server.

Deploying DBQ Tool

You must deploy a DBQ application to your Java application server before you can start using DBQ Tool. To connect DBQ Tool to the Document Broker repository, an SSSP (StreamServe Service Provider) application must also be deployed to the Java application server.

Deployment in Control Center

You deploy the web application archive (WAR) files for the DBQ and SSSP applications in Control Center. For detailed information, see *Deploying a web application* in the *Control Center* documentation.

When you add a **DBQ Tool** application called `<Name>` in Control Center, the following WAR files are deployed:

WAR file	Application
`<Name>_dbq.war`	DBQ application
`sssp.war`	SSSP application If there is an SSSP application already deployed to the portal root, this SSSP application is automatically reused.

If Control Center is not available, you can deploy the web applications via the command line utilities. For more information, see the *Command line utilities* documentation.

Multiple application domains

You can link one **DBQ Tool** application to one or several application domains with StreamServer applications that collect documents. Each application domain can be linked to one single **DBQ Tool** application.

When the **DBQ Tool** application is linked to an application domain, and this application domain is linked to a Document Broker repository, the DBQ Tool can run queries against the documents in this Document Broker repository.

Dependencies in webportal.properties file

The dependency between the DBQ application and the SSSP application is defined in a `webportal.properties` file. The file also contains information about the application domains that the web application is linked to.

The `webportal.properties` file is generated when you link the **DBQ Tool** application to an application domain and resides in the following directory:

```
<Portal root>\<Name>_dbq\WEB-INF\spring\properties
```

If you update the linking, the `webportal.properties` file is automatically updated in the `\properties` folders in the portal root or manual deploy path.

Note: Updated files in the manual deploy path must be manually copied to the Java application server.

Service gateway in application domain configuration file

The SSSP application uses web services, hosted by a service gateway, to access the Document Broker repository. The service gateway is defined in the application domain configuration file.

When you link the **DBQ Tool** application to an application domain, the application domain configuration file is automatically copied to the following directory:

```
<TOMCAT_HOME>\webapps\sssp\domains
```

If you update the application domain configuration, the application domain configuration file is automatically updated in the `\domains` folder in the portal root or manual deploy path.

Note: Updated files in the manual deploy path must be manually copied to the Java application server.

Keep customized property files

If you redeploy the WAR files, for example to apply a hotfix, any customizations that you have made to the property files are overwritten.

To keep customizations, you can use an external directory outside the portal root for the property files. For more information, see *Keeping customized configuration files* in the *Control Center* documentation.

Starting DBQ Tool

Note: Because the web browser shares session context between tabs, it is not supported to run DBQ Tool in more than one tab in the web browser.

You access DBQ Tool in your web browser using the following URL:

```
http://<host>:<port>/<Name>_dbq/?domain=<Application domain>
```

Where:

- *<host>* – Is the host of the Java application server.
- *<port>* – Is the port the Java application server listens to (default is 8080).
- *<Name>*`_dbq` – Is the DBQ Java application working directory.
- `domain=`*<Application domain>* – Is the application domain to be accessed. This parameter is required if the DBQ application is connected to several application domains.

For example:

```
http://localhost:8080/MyDBQ_dbq/
```

Creating DBQs

When you create a Document Broker query (DBQ), you must first select which document types to include in the query. To create a more fine-grained DBQ you must also apply a filter based on metadata values.

To create a DBQ

1. Select the document types to include in the query. See *Selecting document types* on page 36.
2. Specify more filter criteria for the query based on metadata values. See *Using the metadata filter* on page 36.
3. Perform a search to list all documents that fulfill the filter specified. See *Performing a search* on page 37.
4. If needed, specify which documents to include in the query. See *Selecting specific documents in the DBQ* on page 38.
5. Generate the DBQ. See *Generating DBQs* on page 38.

Selecting document types

When you create a DBQ you must first use the document types filter to select which document types to include in the query. When you have selected your document types, all documents in the Document Broker repository that belong to the selected document types are available in DBQ Tool.

If you want to create a DBQ that selects all documents that belong to the document types you specify in this filter, the DBQ is now ready to use. If you want to create a more fine-grained DBQ you must also apply a filter based on metadata values. See *Using the metadata filter* on page 36.

To select document types

1. Click **Add**.
2. In the Add Document Types dialog box, select the document types you want to add, click **Add selected** and click **OK**.

To remove document types from the filter

1. Select the check-boxes for all document types you want to remove.
2. Click **Remove Selected**.

Using the metadata filter

In the DBQ you can create a metadata filter that contains rules based on metadata values. The DBQ will in this case only select documents in the Document Broker repository that fulfill the rules specified in the filter.

The metadata available in the filter is the metadata shared by the selected document types including all system metadata. For example:

- If you select a single document type in the document types filter, all metadata defined for this document type is available.

- If you select several document types, only metadata shared by all document types is available.

To create a rule

1. Click **Add**.
2. In **Metadata Group**, select the metadata group that contains the metadata to use. Note that `STATIC` means system metadata.
3. In **Metadata Name**, select the metadata to use in the rule.
4. In **Operator**, select the operator to use in the rule.
5. In **Value**, enter the metadata value to use in the rule.

Adding several rules

You can add several rules to the metadata filter. All rules that you add to the filter must be fulfilled when selecting documents in the Document Broker repository.

To remove rules from the filter

1. Select the check-boxes for all rules you want to remove.
2. Click **Remove Selected**.

Performing a search

When you have created a filter (document types filter and metadata filter), you can click **Search** to display all matching documents in DBQ Tool. The documents are listed in a table at the bottom of DBQ Tool. This table includes the following columns by default:

- **Creation Date** – date and time the document was created.
- **DocTypes** – document type of the document.
- **State** – processing state of the document.

You can add columns for all available metadata to this table if you need to.

To add columns to the document table

1. Click **Columns**. The Show/hide columns dialog box opens.
2. Specify which tables to show and click **OK**.

Note: You must perform a new search for the changes to take effect.

Changing the number of documents displayed on one page

The table is divided into pages, and the default number of documents on each page is 25. To change this number, you must edit the file `dbq.properties` located in:

`<Portal root>\<Name>_dbq\WEB-INF\spring\properties`

For example for Apache Tomcat:

`<TOMCAT_HOME>\webapps\<Name>_dbq\WEB-INF\spring\properties`

Selecting specific documents in the DBQ

The standard process for creating DBQs is to create a generic query where you specify a document type filter and a metadata filter. This type of query is used to select all documents in the Document Broker repository that match the filter criteria.

In some cases you may need a DBQ that selects specific documents in the Document Broker repository. For example, if a print shop has a printer jam, the affected documents need to be resent. In this case you can select the documents in DBQ Tool, and create a query that selects these documents only. Another example is that DBQ Tool can be used by consultants for testing purposes to delete or process a couple of documents.

To create a DBQ that selects specific documents

1. Select the document types to include in the query. See *Selecting document types* on page 36.
2. Specify more filter criteria for the query based on metadata values. See *Using the metadata filter* on page 36.
3. Perform a search to list all documents that fulfill the filter specified. See *Performing a search* on page 37.
4. In the document list, select all documents to include in the query.

Generating DBQs

When you have created your DBQ you must generate the actual DBQ output. You can either:

- Export the DBQ to a file. See *Exporting a DBQ to file* on page 39.
- Submit he query directly from DBQ Tool. See *Submitting a DBQ to a StreamServer application* on page 39.

When you generate the DBQ you must specify which action to perform.

Action	Description
Process	Process the selected documents.
Process & delete	First process the selected documents, and then delete them from the Document Broker repository.
Delete	Delete the selected documents from the Document Broker repository.

Exporting a DBQ to file

You can export a DBQ to file, and deliver the file to a post-processor via the appropriate input connector, for example a Directory input connector.

When you export you have the option to preview the DBQ in the web browser (click **Open**). In some browsers the preview is opened in the same tab as DBQ Tool, and when you click **Back** to return to DBQ Tool all settings are cleared. If you want to preview the DBQ, the recommendation is to save a copy of the DBQ (click **Save**) and then preview the DBQ in a separate browser/tab.

To export a DBQ

1. In DBQ Tool, click **Export**.
2. In the Export dialog box, select wether to include **All documents defined by filter** or **Selected documents only**.
3. Select the appropriate **Action**.
4. If you do ***not*** want to convert date and time values to UTC, clear the **Convert metadata date/time values to UTC** check-box.
5. Click **Export**. You are prompted to save the file.
6. Click **Save**. A file browser opens.
7. Specify the path and filename and click **Save**.

Submitting a DBQ to a StreamServer application

You can submit a query directly from DBQ Tool to a running StreamServer application. The StreamServer application must be configured to retrieve DBQs via a Service Request input connector where you specify a service name. When you specify which StreamServer application to submit to, you must select the corresponding service name.

DBQ Tool does not show any result after the query is submitted. To see if the query was executed correctly, you can examine the log of the StreamServer application to which the query is submitted.

To submit a DBQ

1. In DBQ Tool, click **Submit**.
2. In the Submit dialog box, select wether to include **All documents defined by filter** or **Selected documents only**.
3. Select the appropriate **Action**.
4. If you do ***not*** want to convert date and time values to UTC, clear the **Convert metadata date/time values to UTC** check-box.
5. From the **Connector** drop-down list, select the appropriate **<*Service name*>** and click **Submit**.

Creating PPQs

You can create PPQs using any editor or application that generates XML output, or auto-generate PPQs while processing documents.

In this section

PPQ syntax

The PPQ syntax is illustrated below.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="value" update="value">
 <database>
<jobset sel="property=value">
<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 42 and *Update* on page 43.
- The `<jobset>` element specifies which batch in the repository to select. See *Job selection* on page 44.
- The `<docset>` element specifies which documents in the repository to select. See *Document selection* on page 46.

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 1 Action=process.

Process all documents.

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

Example 2 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>
```

Update

Use the `update` attribute in the `<s-dbs>` element to specify that documents can be processed by several parallel applications. For example, when using different delivery channels.

Attribute value	Description
`all`	This is also the default behavior if the `update` attribute is not used. Documents are processed by one single application at the time, which ensures that processed documents are delivered only once. Documents are reserved during processing, but can be processed by other applications that use `update=none`. Document status during processing is `processing` and after processing `processed`.
`none`	Documents can be processed and delivered by several parallel applications. Documents are not reserved during processing and can be processed by other applications. Document `Status` is not updated and you must use the *UpdatePPDocStatus* scripting function to update it. **Note:** All selected documents are processed, even if they are locked by other applications that use `update=all`.

Example 3 Update=all

Lock documents while processing.

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

Example 4 Update=none

Process all documents, even if they are locked.

```
<?xml version="1.0" encoding="utf-8"?>
<s-dbs action="process" update=”none”>
    <database>
          <docset sel="all"/>
    </database>
</s-dbs>
```

Job selection

The `<jobset sel="property=value">` sections in the PPQ specify which batch jobs in the runtime repository to select.

Job properties		
Property	**Description**	**Examples**
`All`	Select all jobs in the repository.	`sel="All"`

`ID`	Select a job with either of the following:
	• JobSequenceNumber, (numeric values)
	• StoredJobID (GUID), (numeric or string values)
		(JobSequenceNumber)
		`sel="ID=2"`
		(StoredJobID)
		`sel="ID=&quot;D4B5943B-76B9-4426-906E-75316E141B42&quot;"`

Job properties		
Property	**Description**	**Examples**

`DocumentType`	Select jobs connected to a specific document type.
	The document type can be specified by name or GUID.
		(Document type name)
		`sel="DocumentType='Invoice'"`
		(GUID)
		`sel="DocumentType='98A23889-62E7-B718-8842-2607A980C851'"`

The `<jobset>` element can enclose zero, one or multiple `<docset>` elements in the PPQ, see examples below.

Example 5 **<jobset> combined with <docset>**

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

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

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

Operators

For a list of available operators, see *Filter operators* on page 46.

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 47.
- Metadata filters (`metainfo="filter"`). See *Creating metadata filters* on page 49.

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

`Error`	Select documents processed with or without errors.
	Possible values are `0` and `1`.
		(Documents processed without errors)
		`sel="Error=0"`
		(Documents processed with errors)
		`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 6 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 and time format and keywords

Date values must follow the ISO 8601 standard, see an example below.

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

The following keywords can be used together with the `CreationTime` and `ProcessTime` properties.

Keyword	Description	Examples

`today`	Select documents created or updated today or any day before today.	(Documents created today)
		`sel="CreationTime=today"`
		(Documents updated yesterday)
		`sel="ProcessTime=today-1"`
		(Documents created five days ago)
		`sel="CreationTime=today-5"`

`now`	Select documents created or updated during a period of hours and minutes back in time.	(Documents created during the last four hours)
		`sel="CreationTime>=now-4"`
		(Documents updated during the last hour and twenty minutes)
		`sel="ProcessTime>=now-1:20"`

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 7 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.

Auto-generating PPQs

When a StreamServer application processes documents to be stored in a Document Broker repository, it can also auto-generate a PPQ. This PPQ can be used to select and process all documents in a batch and you have the following options:

- Save the PPQ as a file for later processing of documents stored in the Document Broker repository.
- Send the PPQ directly to an input or output connector for immediate processing.

Note: In most cases it is recommended to send the PPQ to a connector, since it gives you more control and additional processing options. See *Scenarios for auto-generated PPQs* on page 50.

Example 8 Auto generated PPQ.

This example shows an auto-generated PPQ file that will process all documents in the stored batch with job 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, open the Runtime Output Connector Settings dialog box for the Document Broker Plus connector.
2. Activate the generic layer and click the **Job End** icon.
3. On the Device Driver Settings tab, enter a PPQ file name and/or the name of the connector. For information about the settings on the tab, see *Connector settings for auto-generated PPQs* on page 51.

Scenarios for auto-generated PPQs

When to use the Job Info Connector setting

- **Store, post-process and deliver** – When you are using, for example, Service Request or HTTP input connectors and want to store, post-process and deliver the documents within the same job.
- **Return information about stored documents** – When you are using, for example, Service Request or HTTP input connectors and want to return information about what was stored to the input connector.

- **Manipulate PPQ before process** – When you want to manipulate the PPQ before processing it or storing it as a file. You can, for example, send the PPQ file to an XMLIN Event and modify it in an XMLOUT Process before it is written to a File output connector.
- **Post-process without scripting or copying files** – When you want to post-process stored documents directly, without scripting or copying files.

When to use the Job Info File setting

- **Keep record of stored documents** – When you only want to keep record of documents stored by a post-processor job.
- **Upgrade existing Project** – When you upgrade an existing project that uses PPQ files and do not want to change that.

Connector settings for auto-generated PPQs

You configure the connector settings for auto-generated PPQs on the Document Broker Plus connector (the Device Driver Settings tab in the Runtime configuration).

Setting	Description
Job Info File	Specifies the path where to save the auto-generated PPQ, as a file, for example: `C:\strs_data\JobInfo\phone.xml` Note that all folders in the path must exist.
Job Info Connector	Specifies the connector in the project to which the auto-generated PPQ is sent. Only one connector can be specified and it can be an input or output connector. You can use static text, a variable or a lookup table to specify the connector. **Note:** When using input queues in the storing job, you cannot use variables in the runtime settings of the connector receiving the PPQ. The reason is that the PPQ is sent after job end when variables are cleared.
Update documents when processing	**Yes** – Documents are reserved and processed by one post-processor at the time. This setting corresponds to PPQ attribute `update` with value `all`, see *Update* on page 43. **No** – Documents can be processed and delivered by several parallel applications. This setting corresponds to PPQ attribute `update` with value `none`, see *Update* on page 43.

Setting	Description
Selection criteria when processing	**jobset** – The PPQ is generated with a `<jobset sel="ID=x">` tag that uses the job ID as main selection criteria together with a `<docset sel="ID >= x AND ID <= y"/>` range. **docset** – The PPQ is generated with one `<docset sel="ID=x"/>` tag for each generated document. **Note:** The **docset** option requires more memory and has more performance requirements than the **jobset** option. The **docset** option is only recommended if the PPQ is modified before processing. For example, by adding `gmi` tags to modify metadata.