Documentum DQL How To Configure

On this page:

Step 1. Launch Aspire and open the Content Source Management Page

Launch Aspire (if it's not already running).

See:

1. Launch Control.
2. Browse to: http://localhost:50505. For details on using the Aspire Content Source Management page, see Admin UI.

Step 2. Add a new Content Source

To specify exactly which shared folder to crawl, we will need to create a new "Content Source".

To create a new content source:

1. From the Content Source, select **Add Source**.
2. Select **Connector**.

Step 2a. Specify Basic Information

In the **General** tab in the **Content Source Configuration** window, specify basic information for the content source:

1. Enter a content source name in the **Name** field.
   a. This is any useful name that you decide is a good name for the source. It will be displayed in the content source page, in error messages, etc.
2. Click on the **Scheduled** list and select one of the following: *Manually, Periodically*, *Daily,Weekly or Advanced.*
   a. Aspire can automatically schedule content sources to be crawled on a set schedule, such as once a day, several times a week, or periodically (every N minutes or hours). For the this tutorial, you may want to select **Manually** and then set up a regular crawling schedule later.
3. Click on the **Action** list to select one of the following: *Start, Stop*, *Pause,* or *Resume.*
   a. This is the action that will be performed for that specific schedule.
4. Click on the **Crawl** list and select one of the following: *Incremental, Full*, *Real Time,* or *Cache Groups.*
   a. This will be the type of crawl to execute for that specific schedule.

After selecting Scheduled, specify the details, if applicable:

- *Manually*: No additional options.
- *Periodically*: Specify the **Run every** option by entering the number of "hours" and "minutes".
- *Daily*: Specify the **Start time** by clicking on the hours and minutes lists and selecting options.
- *Weekly*: Specify the **Start time** by clicking on the hours and minutes lists and selecting options, and then selecting the check boxes to specify days of the week to run the crawl.
- *Advanced*: Enter a custom CRON Expression (e.g. 0 0 0 ? * *)

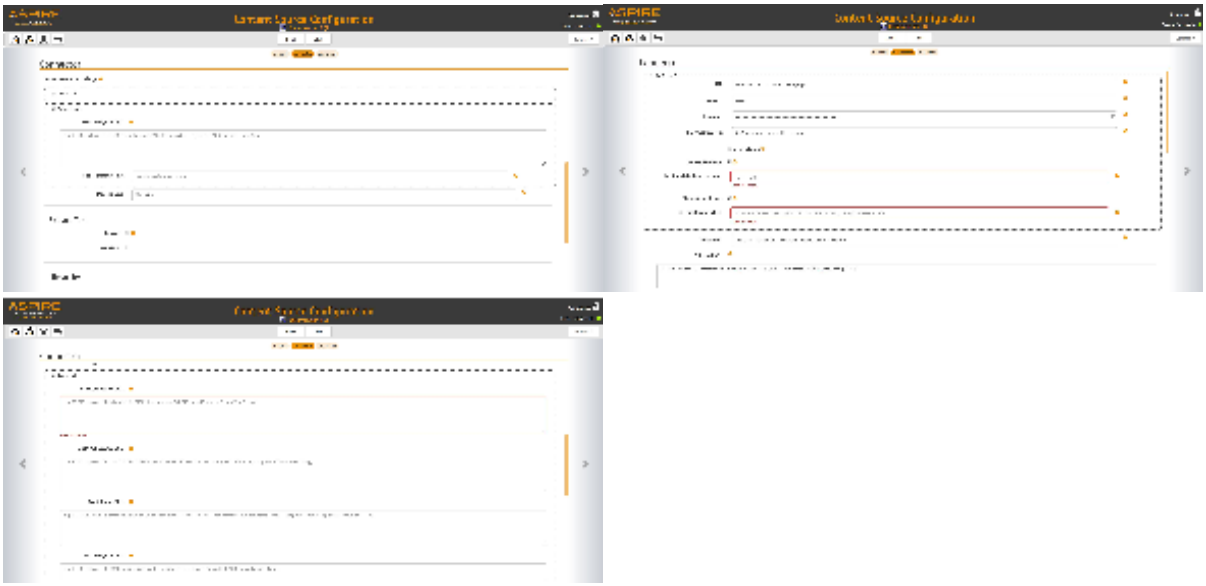
You can add more schedules by selecting the **Add New** option, and rearranging the order of the schedules.

If you want to disable the content source, clear the **Enable** checkbox. This is useful if the folder will be under maintenance and no crawls are desired during that period of time.

Real Time and Cache Groups crawl will be available depending on the connector.

Step 2b. Specify the Connector Information

In the **Connector** tab, specify the connection information to crawl Documentum.



1. Enter the dctm docbase url you want to crawl (Format: dctm://<docbroker-server>:<docbroker-port>/<docbase>.
2. Enter the username (*aspire_crawl_account*).
3. Enter the user's password.
4. Enter the location of the *dfc.properties* file. Make sure the *dfc.properties* file correctly points to the dfc.keystore in the property: *dfc.security.keystore.file*.
5. Check if you want Error Tolerant: Check the option if you want to index only metadata and ignore issues during extract content phase.
6. Use RenditionType option. if selected, you have to provide a list of renditions that you want to index. During fetching the document content the first matching rendition from the list will be used provided the document has this type of rendition. If the document has no rendition type other than the default or doesn't match with any of the specified renditions, the connector will use the default.
7. Metadata attributes option. When checked you have to provide the list of the metadata attributes you want to index. If not selected all document attributes will be used. All those attributes appear in the connector specific part of the indexed document.
8. Enter the webtop URL. This URL will be suffixed with the object ID and used as a value of the displayUrl element in the resulting metadata. For example: http://server-name:port/webtop/component/drl?objectId=
9. Enter a DQL SELECT statement for full crawl. This statement is supposed to query the table with documents - i.e. **dm_document** and provide the set of IDs for further processing.
   You must use exactly these two mandatory fields in SELECT - **r_object_id** and **i_chronicle_id**. There is also the parameter **{SLICES}** to be used as a part of WHERE clause. If used, the full crawl select would be internally run as 16 parallel selects (e.g. <fullSelect> AND r_object_id LIKE '%0' - <fullSelect> AND r_object_id LIKE '%f'). The purpose of this is to provide an option for parallel processing in the scan phase. We index ONLY the current version of the document. This is how it works:
   a. The "id" field in the index is chronicle_id
   b. Full crawl DQL must return one row per document. Something like SELECT r_object_id FROM dm_document.. returns r_object_id for the current version only, while something like select r_object_id from dm_document (all) returns all the versions - but this is FORBIDDEN in our DQL connector
   c. In process stage we use r_object_id for fetching the content stream of the current document version (we also support so called renditions). We also use r_object_id for fetching the object itself using DFC API. We collect then metadata attributes from the fetched object (we can limit the attributes list by appropriate dxf parameter)
10. **Incremental strategy Audit trail**.This strategy uses the same table as full crawl uses for getting the basic set of changes and afterwards it uses dm_audittrail for getting ACL changes and deletes. You enter DQL SELECT statements for the incremental crawl. In those statements, the parameter **${crawlTimeStamp}** must be used in WHERE. This parameter will be expanded by Aspire at the start of crawling. The time of the last crawl start will be used here in the form of Documentum date function - e.g. ${crawlTimeStamp} -> **date('09/22/2016 12:00:00','mm/dd/yyyy hh:mi:ss'):**

a. The incremental crawl DQL SELECT statement is for picking up “adds” and “updates” with the help of the r_modify_time attribute in the main table. The r_object_id and i_chronicle_id fields are the results of SELECT. This query should be exactly the same as the query for the full crawl plus the parameter ${crawlTimeStamp}
b. The Audit ACL DQL SELECT statement for tracking ACL changes. The table for querying is **dm_audittrail**. The fields r_object_id and chronicle_id are the results of SELECT. The parameter ${crawlTimeStamp} must be used in WHERE and also appropriate Documentum event names..
c. The Audit deletes SELECT statement for tracking deletes (documents or versions of documents). The table for querying is **dm_audit trail**. The fields r_object_id and chronicle_id are the results of SELECT. The parameter ${crawlTimeStamp} must be used in WHERE and also appropriate Documentum event names. If document version is deleted the Aspire "update" event will be issued, in case of deleting all versions the Aspire "delete" event will be issued.
d. The Audit safeguard SELECT statement is for checking that the chronicle IDs of documents (retrieved by the use of a previous DQL selects) really exists in the required filter in document table.
In WHERE the parameter **${auditChronicleId}** will be expanded for the chronicle_id value. The field r_object_id is the result of the SELECT. This query also serves as a translator between chronicle_id (used in audit table) and r_object_id (from the document table). In case of deletes all versions of the document this Select returns "nothing" and the index "delete" event will be issued.
e. Select the Delete audit items check box if you want Aspire to delete the already processed row in the audit table.
f. Valid repository time zone id. Not mandatory - UTC being the default value

11. **Incremental strategy Event queue**. This strategy uses Documentum event queue dmi_queue_item for getting all changes.
    a. The DQL connector crawl user should be configured to get the events of his interest. This user should be dedicated only for the purpose of crawling.
    b. DQL connector uses internally for reading events DFC API call “session.getEvents()” – all unread events for the user from previous incremental are fetched and automatically marked as “read”.
    c. The events are internally stored in two groups – deletes + all the others (treated later as add/updates). For each r_object_id only one event is stored in each group.
    d. Add/updates. Because the event contains only r_object_id we need in the main table:
        i. Find the corresponding chronicle_id
        ii. Check if the event belongs to the full crawl filter in order to pick up only events which are of our interest:
            1. So called “safeguard” DQL must be provided with the same filter as in full crawl to check the above mentioned situation. The result is chronicle_id and the **${eventId}** will be expanded for r_object_id from the event. This SELECT also serves as a translator between r_object_id (used in event table) and chronicle_id (from the document table).
            2. This “safeguard approach” has the drawback of reading all enqueued events and filtering them only afterwards.
    e. Deletes
        i. The names of delete events are configurable (the default being dm_destroy, dm_prune). This means that we need here only events which means deleting the whole document - all the versions.
        ii. We simply send all delete events as delete requests to the index. We know that what we actually send is r_object_id but in case of real delete (dm_prune) the r_object_id equals chronicle_id and as a result the real deletion will happen.
        iii. Also we can send sometimes unnecessary delete requests which maybe does not belong to the full crawl filter but this cannot do any harm since we only use chronicle_ids for index id.
12. Enter the max file size. Any file larger than this size will be ignored by the connector. *Unlimited* includes all files.
13. Select other options as needed
    a. **Include/Exclude patterns**: This should be handled in WHERE clause of DQL statements.
    b. **Non-text document filtering**: You should only use the Regex file option for identifying non-text files. The Documentum attribute **a_c ontent_type** is used as a non-text filter field.

Step 2c. Specify Workflow Information

In the **Workflow** tab, specify the workflow steps for the jobs that come out of the crawl. Drag and drop rules to determine which steps an item should follow after being crawled. You can use these rules to specify where to publish the document, or which transformations on the data are needed before sending it to a search engine. See Workflow for more information.

1. For this tutorial, drag and drop the *Publish To File* rule found under the *Publishers* tab to the **onP ublish** Workflow tree.
    a. Specify a **Name** and **Description** for the Publisher.
    b. Click **Add**.
2. Select **Save** and **Done** to return to the Home Page.

Step 3: Initiate a Full Crawl

Now that the content source is set up, the crawl can be initiated.

1. Select the **Full** crawl type option**. (**The default is Incremental.) The first time it will work like a full crawl. After the first crawl,select **Incremental** to crawl for any changes done in the repository.
2. Select **Start**.

During the Crawl

During the crawl, you can do the following:

1. Select **Refresh** on the **Content Sources** page to view the latest status of the crawl.
   The status will show **RUNNING** while the crawl is going, and **CRAWLED** when it is finished.
2. Select **Complete** to view the number of documents crawled so far, the number of documents submitted, and the number of documents with errors.
   If there are errors, you will get a clickable Error that will take you to a detailed error message page.

Step 4: Initiate an Incremental Crawl

1. If you want to process only content updates from the Documentum (documents that are added, modified, or removed), then select **Increment al** instead of **Full**. The connector will automatically identify only changes which have occurred since the last crawl.

   If this is the first time that the connector has crawled, the action of the Incremental option depends on the exact method of *change* discovery. It may perform the same action as a Full crawl and crawl everything, or it may not crawl anything. Thereafter, the Incremental button will only crawl updates.

Statistics are reset for every crawl.

Group Expansion

Group expansion configuration is done on the "Advanced Connector Properties" of the Connector tab.

1. Select the **Advanced Configuration** check box to enable the advanced properties section.
2. Scroll to **Group Expansion** and select the check box.
3. Add a new source for each repository from which you want to expand groups from. (You'll need administrator rights on all of them to be able to do this.)
4. Set the default domain, user name, and password of the crawl account.
5. Set a schedule for group expansion refresh and cleanup.
6. As an optional setting select the **Use external Group Expansion** check box to select an LDAP Cache component for LDAP group expansion. See more info on the LDAP Cache component at LDAP Cache.