Seven Jobs Every Documentum Developer Should Know and Use

M. Scott Roth / Flatirons Solutions Corp.
June 28, 2007

Abstract

Development environments often become corrupted and unusable due to the abuse they endure as temporary and experimental spaces. Documentum provides housekeeping jobs with the Content Server[1] that—if properly used—can prevent development environments from becoming dysfunctional. There are seven jobs that every developer should know and run regularly in their development environment. These jobs ensure your development Docbases stay clean, lean and referentially intact. They can also provide invaluable insight to the content in the Docbase. This paper discusses these seven jobs, makes recommendations regarding their schedule and use, and discusses the value of their capabilities.

[1] The information in this paper is based upon Documentum 5.3 SP3.

Introduction

If you're like me, you're pretty rough on your development environment. Regardless of whether your development environment is physical or virtual, it takes a beating during development. It is often subjected to mass imports and deletes, abandoned workflows, types and DocApps that are altered and re-altered, failed logic and logic unsuccessfully reversed. All these activities build up over time to produce a messy, inconsistent, fragmented, slow and sometimes dysfunctional Docbase. Fortunately, there are some pretty simple things you can do to make sure your development environment stays healthy. The simplest is to run the housekeeping jobs Documentum provides. Most developers probably consider jobs to fall in the Documentum administrator realm, but in a development environment, that responsibility could very well be yours. And even it isn't, it can't hurt to know a little something about housekeeping.

This paper introduces seven jobs that every developer should know about, and provides recommendations on how often they should be run. The seven jobs are:

	Job	Description
1.	DMClean	The DMClean job removes deleted and orphaned objects from the Docbase.
2.	DMFilescan	The DMFileScan job removes deleted and orphaned content files from the file system.
3.	LogPurge	The Log Purge job removes server and session logs from the Docbase and file system.
4.	ConsistencyChecker	The Consistency Checker job runs 77 referential integrity checks on the Docbase.
5	UpdateStats	The Update Stats job updates database table statistics and repairs fragmented tables
6.	QueueMgt	The QueueMgt job deletes dequeued Inbox items from the Docbase.
7.	StateOfDocbase	The State of the Docbase job produces a report of the repository environment and statistics about object types.

There are two reasons why you should not only know about these jobs, but also how to configure and run them:

1. Some of these jobs are not configured to automatically run in the out-of-the-box configuration. Out of the box, Documentum has not enabled any jobs that delete objects or require user customized parameters.

2. The jobs that are configured to automatically run may execute at a time when your development environment is not online (especially if you develop in a virtual environment).

The remainder of this paper will discuss these seven important jobs, what they do, how to run them, and why they are important to you.

All About Jobs

Jobs are programs, or scripts, that run automatically on the Content Server without any user intervention or initiation. These programs are usually diagnostic or housekeeping in nature. Jobs do everything from reporting on free disk space, to synchronizing users and groups with an LDAP server, and from removing orphaned files, to replicating content. In general, a job is scheduled by prescribing the desired day, time and frequency (daily, weekly, monthly, etc.) you would like it to run using the Documentum Administrator (DA) client. A special part of the Content Server, the `agent_exec` process, continually checks the Docbase for jobs that are ready to run (i.e., they are active and their scheduled execution time has arrived).

Implementation

Jobs on the Content Server are implemented as two objects: the job (`dm_job`) and the job method (`dm_method`). The job object holds the scheduling information and job method arguments. The method object holds reference to the actual code that implements the functionality of the job.

Arguments

When the `agent_exec` process launches a job, it passes four standard arguments to the job's method:

Standard Argument	Description
`docbase_name`	Name of the Docbase
`user_name`	Name of the user executing the job (usually the Docbase owner)
`job_id`	ID of the job object
`method_trace_level`	Trace level (default is 0, no trace)

In addition, the method can access additional arguments defined in the job object's `method_arguments` attribute. When the method runs, it uses the `job_id` to retrieve these arguments from the job object. Two common arguments passed using this technique are:

Method Argument	Description
`queueperson`	The user name who should receive email and notifications from the job. If the argument is blank, the job uses the user name defined in the `operator_name` attribute of the server config object (usually the Docbase owner).

Method Argument	Description
`window_interval`	Defines (in minutes) a window on either side of a job's scheduled execution time in which it can run. If the Content Server is down during the scheduled execution time of a job, the `agent_exec` will try to run the job when the Content Server restarts. If the current time is within the window defined by `window_interval`, the job will run. If it is not, it will be rescheduled to run at the scheduled time on the following day/week/month. This variable can be frustrating and confusing because if you manually try to run a job outside of its `window_interval`, the server will not let it run. Because we are discussing a development environment, and running these jobs is more important than performance, I recommend setting this value to 1440 (24 hours). This value guarantees the job will run.

Execution

There are three easy ways to manually run jobs:

1. *Documentum Administrator (DA)*
   - Click Job Management | Jobs.
   - Select a job to run by clicking its selection check box.
   - Choose Tools | Run from the menu.

2. *API Editor*
   - `apply,c,NULL,DO_METHOD,METHOD,S,<job name>, ARGUMENTS,S,'<list of arguments>'`

3. *DQL Editor*
   - `EXECUTE do_method WITH method = '<method name>', arguments = '<list of arguments>'`

Tools and Tips

Finally, here are a few DQL statements and other things you can use to status and manipulate jobs:

- To change the trace level of a job, use DQL:
  - `update dm_job object set method_trace_level = <trace level> where object_name = '<job name>'`

- To determine which jobs are currently running, use this DQL:
  - `select object_name, r_object_id, a_last_invocation from dm_job where a_special_app = 'agentexec'`

- To stop a currently executing job, terminate its process in the Windows Task Manager[2].
  - First, determine the job's process id (PID) with DQL:

    ```
    select object_name, r_object_id, a_last_invocation,
    a_last_completion, a_last_process_id, a_current_status from
    dm_job where a_special_app = 'agentexec' order by
    a_last_process_id
    ```

  - Then, terminate the job's PID in the Window's Task Manager.
  - After you terminate the job, you will need to manually reset the status attribute of the `dm_job` object. Use DQL like this:

    ```
    update dm_job object set a_special_app = '', set
    a_current_status = 'ABORTED', set a_last_completion =
    DATE(TODAY) where object_name = '<job name>'
    ```

- If you find that the DMClean and DMFilescan jobs are running very slowly, one remedy to try is deleting outdated registry keys (This can also speed up UCF downloads if you are experiencing delays with data transfers). The keys to delete are:

  - `HKEY_CURRENT_USER\Software\Documentum\Common\LocalFiles`
  - `HKEY_CURRENT_USER\Software\Documentum\Common\ViewFiles`
  - `HKEY_CURRENT_USER\Software\Documentum\Common\WorkingFiles`

Documentum will automatically recreate these keys when it needs them.

The Seven Jobs

The following sections introduce the seven jobs of interest. Of the seven, four (DMClean, DMFilescan, Log Purge and QueueMgmt) delete obsolete objects from the Docbase and clean up. One runs a battery of integrity checks to ensure your Docbase is referentially healthy (Consistency Checker). Another (Update Stats) updates the statistics associated with each database table and repairs fragmented tables. The last job, State of the Docbase, produces a snapshot report that highlights the configuration and content of the Docbase.

1. DMClean

The first job, DMClean (`dm_DMClean`), is the workhorse of the housekeeping jobs. It looks for and deletes orphaned content objects (`dm_sysobject`), ACLs (`dm_acl`), annotations (`dm_note`) and SendToDistributionList workflow templates. Orphaned objects are objects not referenced by or do not hold reference to any other objects in the Docbase and are just cluttering things up.

[2] I suppose you can use the same methodology with the `ps` and `kill` commands on a UNIX system, though I have not tested it.

By default, the DMClean job is `Inactive` and therefore never runs. In a development environment that is subject to a high degree of volatility, it is critical to run this job to keep the Docbase clean and uncluttered. Therefore, I recommend enabling this job and running it on a weekly basis. In addition to cleaning up the Docbase, running this job will preserve disk space, which can be at a premium in a development or virtualized environment.

When the DMClean job runs, it generates an API script in `%DOCUMENTUM%/dba/log/<docbase ID>/sysadmin` named `<job object ID>.bat` and then executes the script to delete the objects. The deletion of content objects from the Docbase also removes their associated content files on the files system. The job can be configured to generate the script but not run it if you prefer.

Job Name: `dm_DMClean`
Recommended Schedule: Weekly
Method Name: `dm_DMClean`
Method Body: `%DM_HOME%/install/admin/mthd1.ebs`
Method Entry Point `DMClean`
DA Method Arguments:

Argument	Recommended Value	Remarks
`queueperson`		The person to receive notification when the job runs
`clean_content`	`TRUE`	Include content objects.
`clean_note`	`TRUE`	Include note objects
`clean_acl`	`TRUE`	Include ACLs.
`clean_wf_template`	`TRUE`	Include SendToDistributionList workflow templates.
`clean_now`	`TRUE`	TRUE executes the generated API script, FALSE does not.
`clean_castore`	`FALSE`	Clean orphaned objects from Content Addressed (CA) storage.
`window_interval`	`1440`	The window on either side of the scheduled time that the job can run.
`clean_aborted_wf`	`TRUE`	Include aborted workflow templates.

Method

DMClean's method (`dm_DMClean`) code is located in `%DM_HOME%/install/admin/mthd1.ebs`. Yes, this is a Docbasic file. This code will probably be ported to a Java class in the near future, but until then, browse through it. You will discover that DMClean and DMFilescan actually call the same method code. Upon further inspection, you will discover that each of these methods call a built-in utility via the *APPLY* API command. The utility called by DMClean is `%DM_HOME%/bin/dmclean.exe`.

Execution

You can manually execute the DMClean job using the API or DQL like this:

API	`apply,c,NULL,DO_METHOD,METHOD,S,dmclean, ARGUMENTS,S, '-clean_aborted_wf -clean_now'`
DQL	`EXECUTE do_method WITH method = 'dmclean', arguments = '-clean_aborted_wf -clean_now'`

Note that jobs are still constrained by the `window_interval` argument set on the `dm_job` object when run manually.

Outputs

In addition to the API script, the DMClean job generates a report of its activity and saves it to `/System/Sysadmin/Reports/DMClean` in the Docbase. The report is not versioned.

2. DMFilescan

The DMFilescan job (`dm_DMFilescan`) scans the file stores looking for content files that have no associated objects in the Docbase (i.e., the files have been orphaned). It sort of takes the opposite approach from DMClean to achieve a similar result. Like DMClean, when the DMFilescan job runs, it generates a script in `%DOCUMENTUM%/dba/log/<docbase ID>/sysadmin` named `<job object ID>` and executes the script to delete the files. The job can be configured to generate the script but not run it if you prefer.

By default, the DMFilescan job is `Inactive` and therefore never runs. In a development environment that is subject to a high degree of volatility, it is important to run this job to keep the file system clean and reclaim space. If the DMClean job runs regularly, this job does not need to run very often, since DMClean should keep orphaned files under control. That being so, this is a development environment subject to frequent abuse and uncertainty; I therefore recommend enabling this job and running it once a month.

Job Name: `dm_DMFilescan`
Recommended Schedule: Monthly
Method Name: `dm_DMFilescan`
Method Body: `%DM_HOME%/install/admin/mthd1.ebs`
Method Entry Point `Filescan`
DA Method Arguments:

Argument	Recommended Value	Purpose
`queueperson`		The person to receive notification when the job runs
`s`		Storage area to scan (no value indicates scan all).
`from`		Subdirectory from which to start scan.
`to`		Subdirectory in which to end scan.
`scan_now`	`TRUE`	TRUE executes the generated script, FALSE does not.

Argument	Recommended Value	Purpose
force_delete	TRUE	Delete orphaned files younger than 24 hours old.
window_interval	1440	The window on either side of the scheduled time that the job can run.

Method

DMFilescan's method (dm_DMFilescan) code is located in %DM_HOME%/install/admin/mthd1.ebs. This is the same file that contains the DMClean method code. In fact, you will discover that DMClean and DMFilescan call the same method code. Upon further inspection, you will discover that each of these methods call a built-in utility via the *APPLY* API command. The utility called by DMFilescan is %DM_HOME%/ bin/dmfilescan.exe.

Execution

You can manually execute the DMFilescan job using the API or DQL like this:

API	`apply,c,NULL,DO_METHOD,METHOD,S,dmfilescan, ARGUMENTS,S, '-scan_now -force_delete'`
DQL	`EXECUTE do_method WITH method = 'dmfilescan', arguments = '-scan_now -force_delete'`

Note that jobs are still constrained by the window_interval argument set on the dm_job object when run manually.

Outputs

In addition to the API script, the DMFilescan job generates a report of its activity and saves it to /System/Sysadmin/Reports/DMFilescan in the Docbase. The report is versioned.

3. Log Purge

The Log Purge job (dm_LogPurge) deletes system generated log files on the file system and in the Docbase that are older than a user specified age. Specifically, the following eight types of log files are deleted:

Log File	Remark
Server log files	Log files maintained by the Content Server on the file system (%DOCUMENTUM%/dba/log).
Connection broker log files	Log files maintained by the Connection Broker on the file system (%DOCUMENTUM%/dba/log).
Agent Exec log files	Logs the activity of the process responsible for executing jobs on the Content Server (%DOCUMENTUM%/dba/log/<Docbase ID>/agentexec).
Session log files	Log files that record the start and end of every session (%DOCUMENTUM%/dba/log/<Docbase ID>/<user name>).

Log File	Remark
Result log files	Log files generated by methods when their SAVE_RESULT parameter is set to TRUE. These files are saved in the Docbase at /Temp/Result.<method name>.
Job log files	Log files generated by jobs. They are saved in the Docbase at /Temp/Jobs/<job name>.
Job reports	Reports generated by jobs run by the Content Server. These reports are saved in the Docbase at /System/Sysadmin/Reports.
Lifecycle log files	Log files generated by lifecycle operations (e.g., promote, demote). These files are stored on the file system at %DOCUMENTUM%/dba/log/<Docbase ID>/bp and named bp_*.log, depending upon the operation.

By default, the Log Purge job is `Inactive` and therefore never runs. In a development environment, log files can quickly chew up disk space, especially if you run heavily instrumented code or traces for debugging. I recommend you run this job at least monthly to recover disk space.

Job Name: `dm_LogPurge`
Recommended Schedule: Monthly
Method Name: `dm_LogPurge`
Method Body: `%DM_HOME%/install/admin/mthd2.ebs`
Method Entry Point `LogPurge`
DA Method Arguments:

Argument	Recommended Value	Purpose
queueperson		The person to receive notification when the job runs
cutoff_days	30	Log age (delete logs older than this value).
window_interval	1440	The window on either side of the scheduled time that the job can run.

Method

Log Purge's method (`dm_LogPurge`) code is located in `%DM_HOME%/install/admin/mthd2.ebs`. This method is also written in Docbasic. Unlike the previous two methods, this method does not create an API file to do its deleting; it does all the work itself in real time. This method is largely undocumented and not terribly interesting, but it is worth browsing to get a feel for how limiting Docbasic can be.

Execution

You can manually execute the Log Purge job using the API or DQL like this:

API
```
apply,c,NULL,DO_METHOD,METHOD,S,dm_LogPurge,ARGUMENTS,S,
  '-cutoff_days 30'
```

DQL
```
EXECUTE do_method WITH method = 'dm_LogPurge', arguments =
  '-cutoff_days 30'
```

Note that jobs are still constrained by the `window_interval` argument set on the `dm_job` object when run manually.

Outputs

The Log Purge job generates a report of its activity and saves it to `/System/Sysadmin/Reports/LogPurge` in the Docbase. The report is versioned.

4. Consistency Checker

The Consistency Checker job (`dm_ConsistencyChecker`) runs a battery of 77 separate checks on the repository looking for inconsistencies, corruptions and data integrity problems. The job does not fix the problems it finds, but does report the problems using a unique number for each type of error it discovers. The job's report indicates the error number, provides a brief description of each problem, and indicates its severity (Error or Warning).

The specific areas checked and the number of tests run are:

Consistency Checks	Number of Tests
Users and Groups	9
ACLs	14
SysObject	8
Folder and Cabinet	9
Document	3
Content Object	3
Workflow	7
Object Type	4
Data Dictionary	7
Lifecycle	6
Full Text Index	2
Object Type Index	4
Method Object Consistency	1

The Consistency Checker job is configured, out-of-the-box, to run automatically every night at 9:40pm. This is a good thing, provided your Content Server is up and running every night at 9:40pm. If your development environment is virtualized on your local workstation or laptop, and your turn your computer off at night, the Consistency Checker may never get a chance to run. Therefore, adjust the execution time accordingly.

Job Name: `dm_ConsistencyChecker`
Recommended Schedule: Daily
Method Name: `dm_ConsistencyChecker`
Method Body: `%DM_HOME%/install/admin/consistency_checker.ebs`
Method Entry Point `ConsistencyChecker`
DA Method Arguments: None

Method

The Consistency Checker's method (`dm_ConsistencyChecker`) code is located in `%DM_HOME%/install/admin/consistency_checker.ebs`. Yes, this is a Docbasic file also. I suspect this code will be ported to a Java class in the near future, but until then, browse through it. The file contains detailed instructions for adding your own tests to the battery of tests, as well as instructions for running the script manually. I also suggest browsing through the code; it holds a good collection of interesting and useful DQL statements.

Execution

You can manually execute the Consistency Checker job using the API or DQL like this:

API	`apply,c,NULL,DO_METHOD,METHOD,S,dm_ConsistencyChecker`
DQL	`EXECUTE do_method WITH method = 'dm_ConsistencyChecker'`

Note that jobs are still constrained by the `window_interval` argument set on the `dm_job` object when run manually.

The Consistency Checker method can also be run manually and independent of the job. The syntax is:

Cmd	`dmbasic -f%DM_HOME%/install/admin/ consistency_checker.ebs -e Entry_Point -- <repository name> <superuser ID> <superuser password>`

The output is written to standard out (`stdout`) when the method is run from the command line.

Outputs

The Consistency Checker job outputs a report to `/System/Sysadmin/Reports/ConsistencyChecker` in the Docbase. If the job runs successfully, meaning there are no Errors (sever inconsistencies), the report is overwritten. If the job finds Errors (sever inconsistencies) the report is versioned.

5. Update Statistics

The Update Statistics job (`dm_UpdateStats`) scans all the database tables used by the repository and updates the statistics for each table. It will also reorganize tables to optimize performance if it determines that they are fragmented. If you are running Documentum on Oracle or Sybase, the Update Statistics job uses an external file to tweak the query optimizer. The file is `%DOCUMENTUM%/dba/config/<docbase name>/custom_<database name>_stat.sql`. You can add additional commands to this file if you know what you are doing, if you don't know what you are doing, you can negatively affect query performance.

The Update Statistics job is configured, out-of-the-box, to run automatically once a week at 8:30pm. This is a good thing, provided your Content Server is up and running when the job is scheduled to run. If your development environment is virtualized on your local workstation or laptop, and your turn your computer off at night, the Update Statistics may never get a chance to run. Note that this job is CPU and disk-intensive; adjust the execution time accordingly.

Though the job is configured to run out-of-the-box, it is configured in READ mode, meaning the job generates a report, but the statistics are not actually updated and the tables are not reorganized. To take full benefit of this job, it should be run in FIX mode.

Job Name: `dm_UpdateStats`
Recommended Schedule: Weekly
Method Name: `dm_UpdateStats`
Method Body: `%DM_HOME%/install/admin/mthd4.ebs`
Method Entry Point `UpdateStats`
DA Method Arguments:

Argument	Recommended Value	Purpose
`queueperson`		The person to receive notification when the job runs
`server_name`		The name of the database server. The documentation says this is a required parameter for SQL Server and Sybase installations. However, reviewing the code does not obviously reveal where this parameter is ever used.
`dbreindex`	`FIX`	Setting this parameter to FIX will cause the job method to update statistics and reorganize fragmented tables. Setting it to READ will only produce a report.
`window_interval`	`1440`	The window on either side of the scheduled time that the job can run.

Method

The Update Statistics' method (`dm_UpdateStatistics`) code is located in `%DM_HOME%/install/admin/mthd4.ebs`. This method is a really interesting method that contains some pretty cool SQL statements. It is sufficiently documented, so I encourage you to browse through it to understand how the statistics are updated.

Execution

You can manually execute the Update Statistics job using the API or DQL like this:

API	`apply,c,NULL,DO_METHOD,METHOD,S,dm_UpdateStats,ARGUMENTS, S, '-dbreindex FIX'`
DQL	`EXECUTE do_method WITH method = 'dm_UpdateStats', arguments = '-dbreindex FIX'`

Note that jobs are still constrained by the `window_interval` argument set on the `dm_job` object when run manually.

Outputs

The Update Statistics job generates a report of its activity and saves it to `/System/Sysadmin/Reports/UpdateStats` in the Docbase. The report is versioned.

6. Queue Management

The Queue Management job (`dm_QueueMgt`) deletes dequeued Inbox items (`dmi_queue_item`). Tasks in a user's Inbox are marked for deletion and dequeued whenever they are forwarded, completed or deleted from the Inbox. However, the underlying objects in the Docbase are not really deleted. In a development environment--that may include the testing of workflows and other activities that generate Inbox items--the build up of undeleted objects can impact performance.

The Queue Management job deletes queue items based upon age of the objects and a custom DQL predicate passed to it. The job automatically creates a base predicate of `delete_flag = TRUE AND dequeued_date = value(<cutoff_days>)`. Any custom predicate provide is ANDed to the base predicate.

By default, the Queue Management job is `Inactive` and therefore never runs. I recommend you run this job at least weekly (perhaps daily depending upon your application and use of queue objects). Running this job will help keep your Inbox performing well.

Job Name: dm_QueueMgt
Recommended Schedule: Weekly
Method Name: `dm_QueueMgt`
Method Body: `%DM_HOME%/install/admin/mthd2.ebs`
Method Entry Point `QueueMgt`
DA Method Arguments:

Argument	Recommended Value	Purpose
`queueperson`		The person to receive notification when the job runs
`cutoff_days`	`30`	Queue item age—delete objects older than this value.
`custom_predicate`		Additional qualifiers ANDed to the base predicate.
`window_interval`	`1440`	The window on either side of the scheduled time that the job can run.

Method

QueueMgt's method (`dm_QueueMgt`) code is located in `%DM_HOME%/install/admin/mthd2.ebs`. This method is also written in Docbasic and largely undocumented. This method is straightforward in its construction and execution and therefore not particularly interesting as far as examining the code for tips and tricks.

Execution

You can manually execute the LogPurge job using the API or DQL like this:

API	`apply,c,NULL,DO_METHOD,METHOD,S,dm_QueueMgt,ARGUMENTS,S, '-cutoff_days 30'`
DQL	`EXECUTE do_method WITH method = 'dm_QueueMgt', arguments = '-cutoff_days 30'`

Note that jobs are still constrained by the `window_interval` argument set on the `dm_job` object when run manually.

Outputs

The Queue Management job outputs a report to `/System/Sysadmin/Reports/QueueMgt` in the Docbase. The report is versioned.

7. State of the Docbase

The final job, State of the Docbase (`dm_StateOfDocbase`), generates a report covering ten essential areas of a Docbase's configuration and state. This report comes in very handy when troubleshooting problems, preparing for migrations or if you just want to review the status of the Docbase. The ten areas covered by this report are:

1. Docbase configuration (from the docbase config object)
2. Server configuration (from the `server.ini` file)
3. OS, RDBMS, and environment info
4. Registered tables
5. Types
6. Formats
7. Storage info
8. Rendition info
9. Users and groups
10. ACLs

The State of the Docbase job is configured, out-of-the-box, to run automatically every night at 8:45pm. This is a good thing, provided your Documentum Content Server is up and running every night at 8:45pm. If your development environment is virtualized on your local workstation or laptop, and your turn your computer off at night, the State of the Docbase may never get a chance to run. Adjust the execution time accordingly.

Job Name: `dm_StateOfDocbase`
Recommended Schedule: Nightly
Method Name: `dm_StateOfDocbase`
Method Body: `%DM_HOME%/install/admin/mthd4.ebs`
Method Entry Point `StateOfDocbase`
DA Method Arguments:

Argument	Recommended Value	Purpose
`window_interval`	`1440`	The window on either side of the scheduled time that the job can run.

Method

The State of the Docbase's method (`dm_StateOfDocbase`) code is located in `%DM_HOME%\install\admin\mthd4.ebs`. This is a Docbasic file that contains a collection of methods used by the Content Server. Though most of this code is undocumented, browsing through it can be enlightening.

Execution

You can manually execute the LogPurge job using the API or DQL like this:

API	`apply,c,NULL,DO_METHOD,METHOD,S,dm_StateOfDocbase`
DQL	`EXECUTE do_method WITH method = 'dm_StateOfDocbase'`

Note that jobs are still constrained by the `window_interval` argument set on the `dm_job` object when run manually.

Outputs

The State of the Docbase job outputs a report to `/System/Sysadmin/Reports/StateOfDocbase` in the Docbase. The report is always versioned.

Conclusion

To recap, there are seven important jobs that you, as a developer, should have configured and running in your development environment. These jobs work to keep your Docbase lean and clean, and can alert you to any developing problems before they get out of control.

Four of these jobs (DMClean, DMFilescan, LogPurge and QueueMgt) delete obsolete objects from the Docbase and clean up. The Consistency Checker runs a battery of integrity checks to ensure your Docbase is referentially healthy. The last job, State of the Docbase, produces a snapshot report that highlights the configuration and content of the Docbase.

I bring these jobs to your attention for two reasons:

- First, I've been there. I have trashed my development environment in the middle of an iteration and had to rebuild it. This activity can set you back a day or two and frustrate you to no end. Often these situations can be avoided by keeping your environment healthy by running the housekeeping and maintenance jobs discussed here.
- Second, four of these jobs are not configured to run in the out-of-the-box configuration, so to benefit from their capabilities you must enable them. Additionally, enabling them is not enough. You need to configure their parameters, and most importantly, schedule them to run when your development environment is up and running.

For more information about these and other jobs, I encourage you to read the *Content Server Administrator's Guide* and visit the Documentum Developer Forums on the EMC Developer Network (EDN).

About the Author

M. Scott Roth is the technical lead for the Washington, DC office of Flatirons Solutions Corp. He is also the author of the book, *A Beginners Guide to Developing Documentum Desktop Applications*, and the open source Documentum command line client, DOCS.

Thanks to all the FSC engineers who knowingly and unknowingly have contributed to this article!

<SDG><