

End User License Agreement

END-USER LICENSE AGREEMENT FOR ToolSS ("SOFTWARE PRODUCT"). IMPORTANT PLEASE READ THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT CAREFULLY BEFORE CONTINUING WITH THIS PROGRAM INSTALL: SS Solutions & Consulting LLC End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) for the SS Solutions & Consulting LLC software product(s) identified above which may include associated software components, media, printed materials, and "online" or electronic documentation ("SOFTWARE PRODUCT"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. This license agreement represents the entire agreement concerning the program between you and SS Solutions & Consulting LLC, (referred to as "licensor"), and it supersedes any prior proposal, representation, or understanding between the parties. **If you do not agree to the terms of this EULA, do not install or use the SOFTWARE PRODUCT.**

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

1. GRANT OF LICENSE.

The SOFTWARE PRODUCT is licensed as follows:

(a) Installation and Use.

SS Solutions & Consulting LLC grants you the right to install and use copies of the SOFTWARE PRODUCT on your computer running a validly licensed copy of the system for which the SOFTWARE PRODUCT was designed [e.g., Windows 95, Windows NT, Windows 98, Windows 2000, Windows 2003, Windows XP, Windows ME, Windows Vista, EMC Documentum]. All other brands and product names are trademarks of their respective owners.

(b) Backup Copies.

You may also make copies of the SOFTWARE PRODUCT as may be necessary for backup and archival purposes.

2. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.

(a) Maintenance of Copyright Notices.

You must not remove or alter any copyright notices on any and all copies of the SOFTWARE PRODUCT.

(b) Distribution.

You may not distribute registered copies of the SOFTWARE PRODUCT to third parties. Evaluation versions available for download from SS Solutions & Consulting LLC's websites may be freely distributed.

(c) Prohibition on Reverse Engineering, Decompilation, and Disassembly.

You may not reverse engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

(d) Rental.

You may not rent, lease, or lend the SOFTWARE PRODUCT.

(e) Support Services.

SS Solutions & Consulting LLC may provide you with support services related to the SOFTWARE PRODUCT ("Support Services"). Any supplemental software code provided to you as part of the Support Services shall be considered part of the SOFTWARE PRODUCT and subject to the terms and conditions of this EULA.

(f) Compliance with Applicable Laws.

You must comply with all applicable laws regarding use of the SOFTWARE PRODUCT.

3. TERMINATION

Without prejudice to any other rights, SS Solutions & Consulting LLC may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE PRODUCT in your possession.

4. COPYRIGHT

All title, including but not limited to copyrights, in and to the SOFTWARE PRODUCT and any copies thereof are owned by SS Solutions & Consulting LLC or its suppliers. All title and intellectual property rights in and to the content which may be accessed through use of the SOFTWARE PRODUCT is the property of the respective content owner and may be protected by applicable copyright or other intellectual property laws and treaties. This EULA grants you no rights to use such content. All rights not expressly granted are reserved by SS Solutions & Consulting LLC.

5. NO WARRANTIES

SS Solutions & Consulting LLC expressly disclaims any warranty for the SOFTWARE PRODUCT. The SOFTWARE PRODUCT is provided 'As Is' without any express or implied warranty of any kind, including but not limited to any warranties of merchantability, noninfringement, or fitness of a particular purpose. SS Solutions & Consulting LLC does not warrant or assume responsibility for the accuracy or completeness of any information, text, graphics, links or other items contained within the SOFTWARE PRODUCT. SS Solutions & Consulting LLC makes no warranties respecting any harm that may be caused by the transmission of a computer virus, worm, time bomb, logic bomb, or other such computer program. SS Solutions & Consulting LLC further expressly disclaims any warranty or representation to Authorized Users or to any third party.

6. LIMITATION OF LIABILITY

In no event shall SS Solutions & Consulting LLC be liable for any damages (including, without limitation, lost profits, business interruption, or lost information) rising out of 'Authorized Users' use of or inability to use the SOFTWARE PRODUCT, even if SS Solutions & Consulting LLC has been advised of the possibility of such damages. In no event will SS Solutions & Consulting LLC be liable for loss of data or for indirect, special, incidental, consequential (including lost profit), or other damages based in contract, tort or otherwise. SS Solutions & Consulting LLC shall have no liability with respect to the content of the SOFTWARE PRODUCT or any part thereof, including but not limited to errors or omissions contained therein, libel, infringements of rights of publicity, privacy, trademark rights, business interruption, personal injury, loss of privacy, moral rights or the disclosure of confidential information.

SS Solutions & Consulting LLC
ToolSS
CreateFolders
Installation and User Guide

End User License Agreement	1
Overview	2
Background	3
Utility Features and Limitations	3
Deployment	4
Spreadsheet Template Usage	5
Execution	6
Spreadsheet Mode	6
Command Line Mode	7
File Preparation	7
Command Line Execution	8
Other Processing Notes	10
Known Issues	10
Single Quotes	10

Overview

The CreateFolders utility was developed to provide Documentum analysts, administrators, or developers with automation for the design and deployment of folder structure within a repository. The CreateFolders utility was developed as a Java class for easy execution as a command line utility, leveraging the Java classes (DFC) provided with Documentum. The CreateFolders class accepts a CSV input file that can be generated using the provided *CreateFolders-Template.xls* spreadsheet. Designers can use the input file template to define the folder structure to meet requirements. Once the specifications are complete, CreateFolders offers two modes for creation:

1. Spreadsheet Mode: The spreadsheet template includes a worksheet with macros that will allow execution of the CreateFolders class from any workstation where JRE, DFC, and CreateFolders are installed.
2. Command Line Mode: A CSV file can quickly and easily be generated from the input file template. This CSV file is then used as input for the CreateFolders class run in command line mode from any workstation where JRE, DFC, and CreateFolders are installed.

The CreateFolders utility processes each record within the input file and creates an object of the defined object type (typically dm_cabinet or dm_folder, but does allow for the creation of

custom subtypes). CreateFolders also assigns the designated Permission Set (a.k.a. ACL); or, optionally allows default inheritance of Permission Set according to system configuration. A log is generated for each execution.

Background

The generation of folder structure (sometimes referred to as taxonomy) is a common requirement for repository design and configuration. The CreateFolders template provides an easy way to define the folder structure design and can be used during requirements gathering and design workshops to finalize the desired structure and object types. Once the structure is defined, you can then also use the spreadsheet to define the security model for the folder structure by specifying the Permission Set for each cabinet or folder.

Utility Features and Limitations

The following provides a brief summary of the functionality for the utility:

1. CreateFolders template spreadsheet provides for input of cabinet/folder structure. For each defined folder, spreadsheet provides for input of `r_object_type` (required; typically `dm_cabinet` or `dm_folder`, but custom subtypes can also be specified) and, optionally, Owner Name and Permission Set.
2. The CreateFolders utility currently supports a maximum of ten folder levels below the cabinet level.
3. If no Owner Name is defined for a record, then the repository's default inheritance mode will be utilized for assignment of `owner_name` (user account running the utility).
4. If no Permission Set is defined for a record, then the repository's default inheritance mode (e.g. USER, TYPE, FOLDER) will be utilized for security assignment on created folders.
5. If a Permission Set is defined for a record, it *must* be a System Permission Sets (i.e. owned by repository owner). The utility does not support use of internal, custom, or user Permission Sets. Support for Permission Set Templates has not been tested.
6. Specified Permission Sets *must* exist. CreateFolders does not create Permission Sets.
7. Folder names cannot contain a forward slash (/) character. This is a system limitation, as this is used as a path delimiter within Documentum.
8. Folder names cannot end in a trailing space character. Although DCTM clients will allow this, it appears to be a DFC limitation as shown in the example below:
[DM_FOLDER_E_TRAILING_SPACE_IN_NAME]error: The folder name 'Approvals ' contains trailing space(s)
9. If the utility encounters a folder that already exists, it skips processing of that record and posts a warning. Note: it does not verify or update owner name or security for this case. This also allows for reprocessing of the input file if any exceptions are encountered, once the exceptions are resolved.
10. The utility will never delete folders; it will only add folders or skip folders if they already exist.

11. If an exception is encountered for a folder, then an error is posted to the log. All subsequent subfolders of that folder will also fail. In such cases, it is necessary to correct the exception and re-run the input file.
12. Command line Java utility that will accept parameters for username, password, repository name, and input file.
13. The input file must be in CSV format (not XLS). It is necessary to use Save As from Excel to convert to CSV format.
14. The input file must include the header row; this is skipped during processing.
15. Utility can be run from any workstation with JRE and DFC installed. The CreateFolders JAR is compiled for both Java 1.4 (DCTM 5.3) and 1.5 (DCTM 6.0, 6.5) compatibility.
16. Establish a session for the user in the repository as supplied by command line parameter. The user is expected to have at least CABINET privilege in the system if any dm_cabinet will be created. Otherwise, user must have WRITE permission and Change Permissions and Change Location extended permissions in the defined Permission Sets in order to create folders and assign permissions.
17. Process the input file to create cabinets and folder (or custom subtypes), assign object_name, owner_name, permission set, and perform linking to create defined structure.
18. No additional or special handling will be performed for audit trail behavior; standard, default behavior will occur during utility execution.
19. Data Dictionary attribute validation is performed for each created folder. If validation fails, the folder is not created. Note: Typically, there are not attribute validation rules for dm_cabinet and dm_folder, but if custom validation rules are added (or set for custom folder subtypes), they will be enforced.
20. An execution log is created for each execution with a unique file name within the directory from which the utility was executed. The log will minimally include the unique r_object_id and folder path for each object created. Exception messages are posted for records with errors. This report may be used for historical reference.

Deployment

A deployment package consisting of a ZIP archive has been provided that contains the following:

Name	Description
CreateFolders-Template.xls	Input spreadsheet template.
jar	Directory containing Java Archive for deployment to machine where it is desired to run the CreateFolders from. Note: There are two compiles provided, Java 1.4 and Java 1.5.
Installation and User Guide	This document.
ReleaseNotes	Running log of revisions.
Sample	Directory containing sample input and output.

The CreateFolders can be deployed to any machine that has Java JRE (or SDK) and Documentum DFC installed by copying the CreateFolders.jar file to a directory of choice. This could be done on the Content Server, but this is not necessary.

Spreadsheet Template Usage

The *CreateFolders-Template.xls* is a spreadsheet that contains pre-formatted input for the design of folder structure and assignment of object type and permissions.

Tip: It is a good practice to save a backup of the CreateFolders-Template.xls file in a safe location. Additionally, it is a good habit to always use File > Save As immediately after opening the template to save with a new file name. These practices will avoid inadvertent overwriting of the original template file.

The layout of the spreadsheet cannot be modified from its original layout, as the utility is expecting the defined order within the input file. The following shows a simple example:

Cabinet	Folder 1	Folder 2	Folder 3	Folder 4	Folder 5	Folder 6	Folder 7	Folder 8	Folder 9	Folder 10	Owner Name	Permission Set (ACL)	Object Type
Test												dm_acl_superusers	dm_cabinet
	Folder1-Test1											dm_acl_superusers	dm_folder
		Folder2-Test1A										dm_acl_superusers	dm_folder
		Folder2-Test1B										dm_acl_superusers	dm_folder
	Folder1-Test2										workflow_admin	dm_acl_superusers	dm_folder
		Folder2-Test2A									workflow_admin	dm_acl_superusers	dm_folder
		Folder2-Test2B									workflow_admin	dm_acl_superusers	dm_folder

Each column within the spreadsheet represents a folder level. This provides a fairly good representation of the nested folder structure. In some cases, it may be desirable to size the columns for improved readability.

Each line within the spreadsheet represents a single folder, and therefore a folder name can only exist in one column per line. Put another way, no line within the spreadsheet should have more than one folder or cabinet name defined.

Each line within the spreadsheet *must* have an Object Type value, which is the *system* name of the object type. The specified Object Type name must *exactly* match the system name. Typically, this would be dm_cabinet or dm_folder. However, this column was added to allow for the support of custom subtypes of dm_cabinet or dm_folder if desired. If no Object Type is specified, an exception will be thrown for that record.

Each line within the spreadsheet may optionally have an Owner Name defined. If specified, this must be a valid, existing owner_name value. If an invalid owner_name value is specified, an exception will be thrown for that record. If not specified, the owner_name will be assigned according to system default behavior; typically the user that is running the utility.

Each line within the spreadsheet may optionally have a Permission Set defined. If specified, this *must* be a *System* Permission Set (i.e. owned by repository owner). The utility does not support the use of internal, custom, or user Permission Sets. Permission Set Templates (PST) have not been tested. It is believed that a PST may be specified, as long as a default Alias Set can be

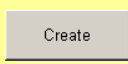
resolved at runtime. See the System Administrators Guide for additional information on PSTs and Alias Set resolution. The specified Permission Set must exist in the repository; the utility does not create Permission Sets. If the specified Permission Set does not exactly match the name of a System Permission Set in the repository, an exception will be thrown. If a Permission Set is not specified, then the default inheritance (e.g. USER, TYPE, FOLDER) defined within the repository will be used to assign a Permission Set to any created cabinets or folders.

Execution

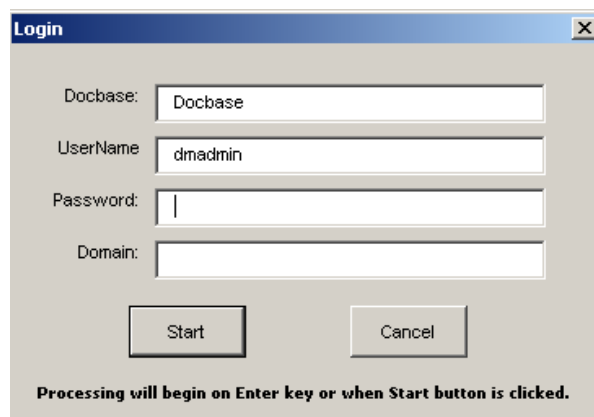
Once folder input data is complete, CreateFolders provides two modes for execution: Spreadsheet Mode and Command Line Mode.

Spreadsheet Mode

In spreadsheet mode, the CreateFolders class can be invoked directly from the spreadsheet on any workstation where JRE, DFC, and CreateFolders are installed. Once the folder configuration input is completed, CreateFolders can be invoked from the Config&Run worksheet. A sample of the Config&Run worksheet is provided below:

	A	B	C	D
1	ToolSS Install Path	C:\ToolSS\Create Folders	Required	Full path specification where CreateFolders.jar is deployed.
2	CLASSPATH	C:\ToolSS\Create Folders\CreateFolders.jar;%CLASSPATH%	Required	Default uses ToolSS Install Path and system configuration.
3	Temp Path	C:\Temp Test	Required	Path where generated input and output files will be saved.
4	Docbase Name	Docbase	Optional	Default for login dialog. Case sensitive.
5	User Name	dmadmin	Optional	Default for login dialog. Case sensitive
6				
7		Click Create button for login dialog. Clicking Start button on subsequent login dialog begins processing.		

Enter the required configuration information. When ready to being processing, click the Create button. This will present a login dialog:



The Login dialog box contains the following fields and controls:

- Docbase:** A text field with the value "Docbase".
- UserName:** A text field with the value "dmadmin".
- Password:** A text field with a single vertical line indicating a password prompt.
- Domain:** An empty text field.
- Buttons:** "Start" and "Cancel" buttons at the bottom.
- Footer:** A message stating "Processing will begin on Enter key or when Start button is clicked."

The Docbase and UserName fields are populated by default based on the entries in the Config&Run worksheet (if entered); otherwise they must be entered manually. Enter password. Use the Enter key or Start button to begin processing.

Note: no further warnings are provided once Enter key or Start button are activated.

When processing is started a command window will display during CreateFolders execution. When processing is complete, a summary of the execution will be presented, such as the following:

```
CreateFolders Started 04-03-2010 11:03:28
Session started in repository DCTM65for user dmadmin.

Processing record 1: Cabinet,Folder 1,Folder 2,Folder 3,Folder 4,Folder 5,Folder 6,Folder
7,Folder 8,Folder 9,Folder 10,Owner Name,Permission Set (ACL),Object Type
Processing record 2: Test,,,,,,,,,dm_acl_superuser,dm_cabinet
Processing record 3: ,Folder1-Test1,,,,,,,,,dm_acl_superuser,dm_folder
Processing record 4: ,,Folder2-Test1A,,,,,,,,,dm_acl_superuser,dm_folder
Processing record 5: ,,Folder2-Test1B,,,,,,,,,dm_acl_superuser,dm_folder
Processing record 6: ,Folder1-Test2,,,,,,,,,workflow_admin,dm_acl_superuser,dm_folder
Processing record 7: ,,Folder2-Test2A,,,,,,,,,workflow_admin,dm_acl_superuser,dm_folder
Processing record 8: ,,Folder2-Test2B,,,,,,,,,workflow_admin,dm_acl_superuser,dm_folder

CreateFolders Finished 04-03-2010 11:03:50
Records Processed: 8
Records Successful: 7 [including header]
Records with Errors: 0
Records with Warnings: 1
See report for details: C:\Temp Test\CreateFolders-2010-04-03-11-03-28.log
```

At completion, three files will be found in the defined Temp Path:

```
CreateFolders-<yyyy-mm-dd-hh-mi-ss>.csv [Generated Input File]
CreateFolders-<yyyy-mm-dd-hh-mi-ss>.out [Execution Summary]
CreateFolders-<yyyy-mm-dd-hh-mi-ss>.log [CreateFolders Execution Log]
```

The CreateFolders Execution Log is described in the next section.

Command Line Mode

Command Line mode allows for preparation of the input file separately from execution of the CreateFolders class. For example, the input file can be prepared from any workstation, and then used as input for running on another workstation where DFC and CreateFolders are deployed. Thus, this is a two step process: File Preparation and Command Line Execution.

File Preparation

Once the spreadsheet is configured as desired and reviewed and approved, it must be converted to CSV format for use with the CreateFolders utility via Command Line Mode. If no modifications to the layout have been made and no additional data exists to the right or below input data, then the file can be converted via the File > Save As > CSV menu option within Excel.

Alternatively, the following is the recommended approach for preparing CSV files for the CreateFolders class. This approach helps to avoid input data processing issues. This will also avoid having to save the original spreadsheet as CSV (which can lead to confusion if modified after saving as CSV).

1. Select upper left-most cell (A1).
2. <shift> select lower right-most cell with data (e.g. the Object Type column for the last row of data). Desired data should now be highlighted.
3. Edit > Copy.
4. Open new worksheet.
5. Select upper left-most cell (A1).
6. Edit > Paste.
7. File > Save As > CSV. (ignore warnings about format and force the save).
8. Close the new worksheet (ignore any warnings about saving changes at this point).
9. Copy CSV to desired location for execution of CreateFolders utility.

Tip: After the input file is run through the CreateFolders class, rename it using the same name as generated for the utility report (with timestamp). Or, vice-versa, rename the report to the same name as the input file (but retaining the .log extension). With either approach, it makes it easier to tell which input file went with which report.

Command Line Execution

CreateFolders is a command line utility. Parameters are passed as space-separated arguments using the following generic syntax for the class:

```
CreateFolders [docbase] [username] [password] [input file spec]
```

The parameters must be passed in the order specified and described below:

Order	Name	Description
1	docbase	Name of the repository. Note: Case sensitive
2	username	Username for repository session.
3	password	User's password for repository session. Note: not masked.
4	input file spec	The full file path and name specification for the input file. Note: if the path or name contains spaces, this must be enclosed in double quotes.

Example generic syntax for connection to TEST repository as MyUser for input file named myinput.csv in C:\temp:

```
CreateFolders TEST MyUser ***** C:\temp\myinput.csv
```

In order for the CreateFolders class to function properly, it may be necessary to add classpath information to the command line syntax. This depends on your user and machine configuration. Minimally, the tool must be able to resolve the classpath to:

- Standard Java classes in installed JRE or JDK (typically, these would be available via your system or user CLASSPATH configuration)
- dfc.jar (typically configured under DCTM shared directory)
- CreateFolders.jar (directory where this utility was deployed to)

The following provides an example of the full command line syntax (Windows) when running from the directory where the CreateFolders.jar file was deployed to, assuming JRE and DCTM resources are available via CLASSPATH environment variable:

```
java -classpath .;\CreateFolders.jar;%CLASSPATH% com.sssc.toolss.cf.CreateFolders TEST MyUser
***** C:\temp\myinput.csv
```

The following shows a similar example for UNIX:

```
java -classpath ../CreateFolders.jar:$CLASSPATH com.sssc.toolss.cf.CreateFolders TEST MyUser
***** /temp/myinput.csv
```

Note: if a java.lang.NoClassDefFoundError is returned when attempting to run the utility, it is most likely a classpath configuration issue.

Note: if a parameter contains a space character, it must be enclosed in double quotes.

Once the utility begins execution, information will be displayed via standard output (e.g. command console). A line is printed for each input record processed, so you can track execution progress. At the conclusion of execution, a summary of information is displayed similar to the following:

```
CreateFolders Finished 04-03-2010 11:03:50
Records Processed: 8
Records Successful: 7 [including header]
Records with Errors: 0
Records with Warnings: 1
See report for details: C:\Temp Test\CreateFolders-2010-04-03-11-03-28.log
```

Additionally, an execution log is generated in the directory from where the utility was initiated at the command prompt. It will be named using the following convention:

CreateFolders-MM-DD-YYYY-HH-mm-ss.log

This naming convention will ensure a unique filename for each execution. This log will contain:

1. Header section with the class name, start time, and input info.
2. Record section with metadata for all input records processed.
3. Summary section with stop time and summary counts (processed successful, errors, warnings).

The Record section contains the following information in CSV format (this allows opening of the reports in MS Excel [by changing the extension to .csv] for easy review and manipulation in spreadsheet format):

"Record Count","Status","Folder Path","Message"

Record Count	Running count of input records processed.
Status	INFO: Used for header row only (skipped). SUCCESS: folder created successfully.. WARNING: specified folder already exists and skipped. ERROR: Failed to create folder due to exception.
Folder Path	The full folder path for the input record. N/A for header record.
Message	If Status = SUCCESS, then the r_object_id of the folder. If Status = INFO, then "Skipping header record." If Status = WARNING, then " Folder already exists; record skipped. ID = <r_object_id>" If Status = ERROR, then Exception message.

Other Processing Notes

- The CreateFolders utility only creates folders, as its name implies. It will never delete folders. Thus, the utility does not perform any checking or cleanup to ensure that any extra folders that are not specified in the input file do not exist. For example, if a cabinet named Test already exists with three subfolders named Test1, Test2, and Test3, and the input file has folder structure defined for Test cabinet with three subfolders Test4, Test5, and Test6; then, at the end of execution the Test cabinet will have all six subfolders. The CreateFolders utility would only add the folders defined in the input file because they do not already exist. It does not check or attempt to remove the previously existing folders Test1, Test2, and Test3 even though they are not specified within the input file. Therefore, it is recommended to review what exists within the repository before running the utility. In some cases, it may be desirable to move any existing folders from similar structure defined within the input file (such as Test1, Test2, Test3 folders) to ensure that you wind up with only the folder structure defined within the input file.

Known Issues

Single Quotes

During testing, it was discovered that the utility does not support the use of single quote characters within folder names, although this is permitted within a repository. Typically the use of single quote characters within names are discouraged, as this can lead to query and processing errors. If single quotes are used, the following exception will be encountered:

Problem processing folder record: [DM_API_E_QUERY_FAIL]error: Query failed:
select r_object_id from dm_folder where any r_folder_path = '<your path with single
quote>'

Support for single quotes may be added as a future enhancement. Until then, it is necessary to remove any single quotes.