**USE CASES**

**FOR**

**LIGHT WEIGHT PUBLISHING**

****

**3750 Zanker Road**

**San Jose, CA. 95134**

Table of Contents

[1 Introduction 2](#_Toc454391118)

[2 Functional Scope 2](#_Toc454391119)

[3 Actors 2](#_Toc454391120)

[4 Use Cases 2](#_Toc454391121)

# Introduction

This document describes the use case scenarios for the Light Weight Publishing (LWP) RESTful web services that allow clients to perform content upload/update/activate/deactivate functionalities remotely that do not need full-fledged CMS functionality but need a mechanism to publish the content under www.cisco.com WEM capabilities in their own applications.

LWP is a parallel pipe for transporting large blobs/binaries directly to WEM web server’s cache. It creates a metadata node in AEM author/publish instances to support metadata related functionality and content access control needs.

# Functional Scope

Here is the proposal for August release scope:

Any standalone DAM assets that doesn't require authoring and/or language translation capabilities are the candidates for LWP.

Potential Clients for LWP after August release with limited scope.

1. Thumbnails of marketing library.

2. Field Notices (TBD) - Right candidate.

3. Sales connect to upload SPA assets from GIT.

4. IT Commerce team FTP client - Images upload to Documentum. (Contact Pragyanam/Aparna Bomma)

# Actors

* User : A logged in/authorized client
* Client : Http Client, SoapUI, User Interface
* Sudo Users : wemadm/xxwem for content upload, activate and deactivate
* Admin : Role based authentication for service deployments

# Use Cases

|  |  |
| --- | --- |
| UC-01 | Upload DAM content that doesn't require authoring and/or language translation capabilities. |
| Description | A user can upload/activate assets on DAM area |
| Preconditions | A User has a user account in LDAP and the user have the required permissions setup in WEM system  The clients should be able to send inputs and consume outputs in JSON format. |
| Actors | User, client and sudo users |
| Normal Sequence | 1. The user request the “uploadcontent” service through the supported clients with valid credentials along with required inputs in JSON format. 2. Client application sends an HTTP POST request along with file’s binary data and Meta data in JSON format. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system creates the node in DAM area and based on inputs specified in JSON. 5. The system upload/copy documents (with sudo user xxwem) into author web server cache if node creation is successful in DAM area.    * If segment path given in the input json is other than guestandabove, system maintains authentication file in webserver cache. Ex: **testdocument.docx** is the input file. **testdocument.\_\_AuthChck\_\_.docx**    * If segment path given in the input json is guestandabove, system maintains two copies of documents in webserver cache in the given format :   **testdocument.docx** and **testdocument.\_\_AuthChck\_\_.docx**   1. If “cisco:activateNow” is given in the input JSON, copy documents (with sudo user xxwem) into publish web server cache and the assets and its corresponding pages will be activated once it passes through all the validations.   Zip Upload:   1. The user request the “uploadcontent” service through the supported clients with valid credentials along with required inputs in JSON format. 2. Client application sends an HTTP POST request along with zip file’s binary data and Meta data in JSON format. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system extracts the zip file and process the each file for creating node in DAM area and based on inputs specified in JSON. 5. The system upload/copy the extracted files (with sudo user xxwem) into author web server cache if node creation is successful in DAM area.    * If segment path given in the input json is other than guestandabove, system maintains authentication file in webserver cache. Ex: **testdocument.docx** is the input file. **testdocument.\_\_AuthChck\_\_.docx**    * If segment path given in the input json is guestandabove, system maintains two copies of documents in webserver cache in the given format :   **testdocument.docx** and **testdocument.\_\_AuthChck\_\_.docx**   1. If “cisco:activateNow” is given in the input JSON, copy documents (with sudo user xxwem) into publish web server cache and the assets and its corresponding pages will be activated once it passes through all the validations. |
| Service URL | /c/services/wem/lwp/uploadcontent |
| Post conditions | The service returns the response in the following JSON format:  {  "statuscode": "200",  "sourcepath": "/content/dam/en/us/td/docs/test.pdf",  "statusmessage": "Content created Successfully"  } |
| Exceptions | The code that represents success or failure of the operation. The status code other than 200 indicates an error.  200 – Successful  400 – Bad request that indicates errors in the inputs sent.  403 – User has no privilege to perform requested operation  500 – Internal system error |
| Comments |  |

|  |  |
| --- | --- |
| UC-02 | Push single page assets into web server cache |
| Description | A user can upload/activate documents into webserver cache. |
| Preconditions | A User has a user account in LDAP and the user should send access token (for authentication) and **cisco:isNonCMSRequest = true** with every request to service. |
| Actors | User, client and sudo users |
| Normal Sequence | 1. The user request the “uploadcontent” service through the supported clients with valid credentials along with required inputs in JSON format. 2. Client application sends an HTTP POST request along with file’s binary data and additional information in JSON format. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system upload/copy documents (with sudo user xxwem) into author web server cache if access token is valid. 5. If “cisco:activateNow” is given in the input JSON, copy documents (with sudo user xxwem) into publish web server cache.   Zip Upload:   1. The user request the “uploadcontent” service through the supported clients with valid credentials along with required inputs in JSON format. 2. Client application sends an HTTP POST request along with zip file’s binary data and additional information in JSON format. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system extract the zip file and upload/copy (with sudo user xxwem) all the extracted documents into author web server cache if access token is valid. 5. If “cisco:activateNow” is given in the input JSON, copy documents (with sudo user xxwem) into publish web server cache. |
| Service URL | /c/services/wem/lwp/uploadcontent |
| Post conditions | The service returns the response in the following JSON format:  {  "statuscode": "200",  "sourcepath": "/content/dam/en/us/td/docs/test.pdf",  "statusmessage": "Content created Successfully"  } |
| Exceptions | The code that represents success or failure of the operation. The status code other than 200 indicates an error.  200 – Successful  400 – Bad request that indicates errors in the inputs sent.  403 – User has no privilege to perform requested operation  500 – Internal system error |
| Comments |  |

|  |  |
| --- | --- |
| UC-03 | Update DAM content metadata |
| Description | A user can update/activate the existing content meta data |
| Preconditions | A User has a user account in LDAP and the user have the required permissions setup in WEM system  The clients should be able to send inputs and consume outputs in JSON format. |
| Actors | User, client and sudo users |
| Normal Sequence | 1. The user request the “updatemetadata” service through the supported clients with his credentials along with required inputs in JSON format. 2. Client application sends an HTTP POST request along with file’s binary data and Meta data in JSON format. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system updates the node Meta data in DAM area based on inputs specified in JSON. 5. The system update/rename documents (with sudo user xxwem) which exist in author webserver cache.  * If segment path given in the input json is other than guestandabove, system maintains authentication file in webserver cache. Ex: **testdocument.docx** is the input file. **testdocument.\_\_AuthChck\_\_.docx** * If segment path given in the input json is guestandabove, system maintains two copies of documents in webserver cache in the given format :   **testdocument.docx** and **testdocument.\_\_AuthChck\_\_.docx**   1. If “cisco:activateNow” is given in the input JSON, copy documents (with sudo user xxwem) into publish web server cache and the asset and its corresponding pages will be activated once it passes through all the validations. |
| Service URL | /c/services/wem/lwp/updatemetadata |
| Post conditions | The service returns the response in the following JSON format:  {  "statuscode": "200",  "sourcepath": "/content/dam/en/us/td/docs/test.pdf",  "statusmessage": "Metadata Updated Successfully"  } |
| Exceptions | The code that represents success or failure of the operation. The status code other than 200 indicates an error.  200 – Successful  400 – Bad request that indicates errors in the inputs sent.  403 – User has no privilege to perform requested operation  500 – Internal system error |
| Comments |  |

|  |  |
| --- | --- |
| UC-04 | Activate content |
| Description | The “activatecontent” service is used to publish content from author instance to publish instance and copy large blobs/binaries directly to WEM web server’s cache |
| Preconditions | A User has a user account in LDAP and the user have the required permissions setup in WEM system  The clients should be able to send input path/paths and consume outputs in JSON format. |
| Actors | User, client and sudo users |
| Normal Sequence | Single file Activation :   1. The user request the “activatecontent” service through the supported clients with his credentials along with required input path. 2. Client application sends an HTTP POST request along with file path. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system copy the content and respective renditions from author webserver cache to publish webserver cache and activates the DAM asset along with respective renditions from author repository to publish repository.   Folder Activation:   1. The user request the “activatecontent” service through the supported clients with his credentials along with required input folder path. 2. Client application sends an HTTP POST request along with folder path. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system copy the content/contents under the given folder from author webserver cache to publish webserver cache and activates the DAM asset/assets under the given folder in AEM. 5. The system copy all the renditions of the respective contents under folder will be copied to publish webserver cache.   Multi-files Activation:   1. The user request the “activatecontent” service through the supported clients with his credentials along with required input paths. 2. Client application sends an HTTP POST request along with comma separated file paths. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system copy each input content from author webserver cache to publish webserver cache and activates the respective DAM asset in AEM. |
| Service URL | /c/services/wem/lwp/activatecontent |
| Post conditions | The service returns the response in the following JSON format:  {  "statuscode": "200",  "sourcepath": "/content/dam/en/us/td/docs/test.pdf",  "statusmessage": "Content activation started"  } |
| Exceptions | The code that represents success or failure of the operation. The status code other than 200 indicates an error.  200 – Successful  400 – Bad request that indicates errors in the inputs sent.  403 – User has no privilege to perform requested operation  500 – Internal system error |
| Comments |  |

|  |  |
| --- | --- |
| UC-05 | Deactivate content |
| Description | The “deactivatecontent” service is used to expire published content. This will remove the content from the publish instance. The content still remains in the author instance. And deactivation can also be scheduled at a given date and time. |
| Preconditions | A User has a user account in LDAP and the user have the required permissions setup in WEM system  The clients should be able to send input path/paths and consume outputs in JSON format. |
| Actors | User, client and sudo users |
| Normal Sequence | Single File Deactivation:   1. The user request the “deactivatecontent” service through the supported clients with his credentials along with required input path. 2. Client application sends an HTTP POST request along with input file path to be deactivated. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system removes the content node from aem publish repository and removes the content/blobs/binary data from publish webserver cache.   Folder Deactivation:   1. The user request the “deactivatecontent” service through the supported clients with his credentials along with required input folder path. 2. Client application sends an HTTP POST request along with input folder path to be deactivated. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system removes the content node/nodes under the given folder path from aem publish repository and removes the content/blobs/binary data from publish webserver cache.   Multi File Deactivation:   1. The user request the “deactivatecontent” service through the supported clients with his credentials along with required input path/paths. 2. Client application sends an HTTP POST request along with input file paths to be deactivated with comma separated. 3. The system checks the validity/completeness of data inputs- Ensures the information input into the system is accurate. 4. The system removes the each content node from aem publish repository and removes the content/blobs/binary data from publish webserver cache. |
| Service URL | /c/services/wem/lwp/deactivatecontent |
| Post conditions | The service returns the response in the following JSON format:  {  "statuscode": "200",  "sourcepath": "/content/dam/en/us/td/docs/test.pdf",  "statusmessage": "Content deactivation started"  } |
| Exceptions | The code that represents success or failure of the operation. The status code other than 200 indicates an error.  200 – Successful  400 – Bad request that indicates errors in the inputs sent.  403 – User has no privilege to perform requested operation  500 – Internal system error |
| Comments |  |