opentext™

WebReports Administration

OpenText™ Content Server

This document is part of the Content Server Admin Online Help documentation list. If conflicts exist, the Online Help supersedes this document.

LLESWEBR160205-AGD-EN-01

WebReports Administration
OpenText™ Content Server
LLESWEBR160205-AGD-EN-01
Rev.: 2018-June-05

This documentation has been created for software version 16.2.5.
It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at https://knowledge.opentext.com.

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
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

Administering the WebReports Module

> **Important**
> The WebReports module is installed with Content Server, but is separately licensed. If you find that WebReports features are not available, please contact OpenText Support for information about how to purchase a license for the WebReports module. For more information, see *OpenText Content Server - Administering Content Server (LLESWBA-AGD)*.

A **WebReports Administration** section in the Content Server administration pages has been provided for some of the unique WebReport functions that require settings.

Definitions of Key Terms

The following terms are used in this documentation:

- **System Administrators** are users who manage **WebReports** permissions and access requirements, and who also have access to the LiveReports, and other data sources that supply data to the WebReports. The various tasks and functions described in this document are intended for these administrators.
- **End Users** are Content Server consumers who can see the output of a WebReport.
- **Developers** are users who may be involved in creating a new WebReport or editing an existing one to change the look, feel, and behavior of the WebReport. For any particular WebReport, Developers fall into two categories:
  - Developers who can add or edit a version of the reportview.
  - Developers who can only retrieve or download an existing reportview. These developers cannot change a reportview, but can reuse and existing one when generating a new WebReport.
- A **Data source** is any Content Server object that can be used to generate a data set for a WebReport. This is most commonly a LiveReport, but could also be a saved search query, a form template, or a form.
- A **LiveReport** is a Content Server item that uses SQL statements to communicate directly with the database to obtain statistical information from, or potentially modify, the Content Server database. The Content Server administrator may create new LiveReports, and grant other users permission to run them. Each user can see the LiveReports for which they have permissions in their Personal Workspace, on the **LiveReports** tab on the **Reports** page.
- A **Reportview** is the template used as the starting point for generating a WebReport object. Each reportview may contain traditional Web code, such as HTML, as well as custom tags that are used to determine where the data source data fields will be inserted in the output Web page.

A **Reportview** is the document that is stored for each WebReport object. These reportviews contain traditional Web code, such as HTML, as well as custom tags that are used to determine where the data source data fields will be inserted in the output Web page.

- **System Administrators** are users who manage **WebReports** permissions and access requirements, and who also have access to the LiveReports, and other data sources that supply data to the WebReports. The various tasks and functions described in this document are intended for these administrators.

1.1 Providing Access to WebReports

Starting from the 10.5.0 2015-09 Patch and later, *new* installations of Content Server will, by default, restrict access to WebReports. *Existing* installations of Content Server that are patched with the 2015-09 Patch, will continue to provide all users with the same open access to WebReports as before the patch.

For new installations of Content Server that were updated to the 2015-09 Patch, you must explicitly add each user to the group that is allowed to create WebReports so that the user will be able to see the WebReport object type in the **Add Item** menu.

1.1.1 To Provide Users with Access to the WebReport Object Type

To provide users with access to the WebReport object type:

1. On the **Content Server Administration** page, in the **System Administration** section, click **Administer Object and Usage Privileges**.
2. On the **Administer Object and Usage Privileges** page, in the **Object Privileges** area, scroll to the WebReport object type and click the **Edit Restrictions** link.

   Note: The WebReports module is installed with Content Server, but is separately licensed. If you have not applied your WebReports license to your Content Server installation, the **WebReport** object type will not appear in the list. For more information about how to manage the WebReports license, see "License Key Functions" on page 28.
3. On the **Edit Group: WebReports** page, follow the *OpenText Content Server - Administering Content Server (LLESWBA-AGD)* procedure.

1.2 Using the WebReports Administration Section

The **WebReports Administration** section provides the following settings and options:

- "Flush Cache" on page 8 removes all compiled reportviews from the cache location.
- "Install the Requests.js Library" on page 8 provides an interface to install the Requests Library and associated files to the JavaScript library location.
- "Manage Category Data Source Configuration" on page 9 updates the maximum number of categories and attribute display parameters.
- "Manage Destination Media Types" on page 10 configures the media types that can be used for WebReports output.
- "Manage Search Query Integration" on page 11 manages the ability to invoke WebReports directly from the search screen.
- "Manage Tags and Sub-Tags" on page 13 selectively disables WebReports tags and sub-tags.
- "Manage Trusted Files" on page 14 configures a set of trusted external files for use by WebReports.
- "Manage User/Group WR Triggers" on page 17 determines which User or Group can trigger a WebReport.
- "Manage WebReports Conversion" on page 18 changes the sleep interval for the conversion agent and to set the input and output directories to manage PDF conversion.
- "WebReports Schedules" on page 22changes the sleep interval for the schedule agent, and to enable, disable, or permanently delete individual schedules.
- "Manage WebReports Scripting" on page 24 enables or disables the scripting of an individual WebReport.
- "Managing WebReports Services" on page 25 enables, disables, or restricts the WebReports services feature.
- "Manage WR Triggers " on page 26 determines which node types can trigger a WebReport.
- "Miscellaneous WebReports Settings" on page 27 sets the values of miscellaneous keys in the WebReports section of the `opentext.ini` file.
- "License Key Functions" on page 28 sets or changes the WebReports License Key and display licensing status.
- "WebReports Node Administration" on page 29 identifies and updates reportviews using out-of-date syntax.
- "WebReports Sub-Tag Builder" on page 30 rebuilds all sub-tags including any subtags that have been created in the sub-tags folder.

1.2.1 Flush Cache

The **Flush Cache** page allows you to remove all the compiled reportviews from the cache location. This is useful if you are having problems with certain reportview compilations, because any removed objects will be re-compiled.

Note: This utility currently only flushes the cache for the server on which `rktengine.flushcache` is run.

To Flush the Cache

To flush the cache:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Flush Cache**.

   Note: This automatically starts the process of flushing the cache location.

2. On the **Flush Cache** page, wait until the flush is complete and the number of flushed objects is reported as completed. .

3. Optional Click **Admin Home** to return to the **Content Server Administration** page.

1.2.2 Install the Requests.js Library

The **Install Requests.js Library** page allows you to install the `requests.js` library and associated files to the following JavaScript library location: `/img/webreports/library/`

When the **Install Requests.js Library** page loads it is pre-populated with default values.

To Install the Requests.js Library

To install the request.js library:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Install Requests.js Library**.

2. On the **Install Requests.js Library** page, provide the following values:

   - **URL Prefix Setting**

     Optional. If in doubt, accept the default settings for the current instance.

     - If this is left blank, the installer will include JavaScript code to obtain the values using information available in the URL during loading of the `requests.js` file into the client.

- If you provide a value, the case of the value must match what a user would type, or else it can cause problems with any requests that expect specific responses. For example, if you provide `/CS/cs.exe`, using uppercase "CS", and a user tries to sign in with `/cs/cs.exe`, using (lowercase "cs"), the sign in attempt will fail.

- **Support Directory Image Path**

  Optional. If no value is provided, the installer will include JavaScript code to obtain the values using information available in the URL during loading of the `requests.js` file into the client.

- **Available Source Versions**

  Mandatory. Accept the default value, or select another version for the `requests.js` library.

- **Themes Location Patch**

  Mandatory. Accept the default value, or enter another path.

- **Available Themes**

  Mandatory. Accept the default, pre-populated value, or select another theme

3. Do one of the following:

   - To submit the form and install the `requests.js` library, click **Install**.
   - To clear any changes and restore the default values, click **Reset**.
   - To cancel and return to the main administration page, click **Cancel**.

1.2.3 Manage Category Data Source Configuration

The **Manage Category Data Source Configuration** page allows you to configure some general settings used by reports that use the Category Data Source type. By controlling the output format and settings, you can help ease the performance demands of running the report.

Important

The WebReports module is installed with Content Server, but is separately licensed. If the **Manage Category Data Source Configuration** link does not appear, please contact OpenText Support for information about how to purchase a license for the WebReports module.

If you have purchased and applied your WebReports license and you still cannot see the **Manage Category Data Source Configuration** link, please restart Content Server.

To Manage the Category Data Source Configuration

To manage the category data source configuration:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage Category Data Source Configuration**.
2. On the **Manage Category Data Source Configuration** page, provide the following information:
   - **Maximum Number of Categories**

     Mandatory. Select a number to determine how many categories a user can select to report on

     Note: The larger this number is, the more complex is the query sent to the database in cases where a user selects the maximum number of categories.
   - **Number of Attributes to Display Across**

     Mandatory. Select the number of selectable attributes to present on each *row* on the WebReport data source configuration page (not in the report itself).
   - **Multi-Valued Attribute Delimiter**

     Mandatory. Enter the character used to separate multi-values that are returned by a report.
   - **Maximum Number of Results**

     Mandatory. Enter a number to limit the results returned by all reports using Category as a Data Source. Any any results above that number will be discarded.
3. Click **Submit**.

1.2.4 Manage Destination Media Types

The **Manage Destination Media Types** page allows you to add, edit, and delete the destination media types which a WebReport can user for output. Any existing destination media types appear when the page is loaded. You can change the name of a media type by changing the data in the **Media Type** field.

To Manage Destination Media Types

To manage destination media types:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage Destination Media Types**.
2. On the **Manage Destination Media Types** page, in the **Media Type** list, do one or more of the following:
   - (Optional) If you want to remove a media type, scroll to the row for the media type that you want to remove and click **Delete Row**  .
   - (Optional) If you want to add a media type, scroll to the bottom of the list, click **Add Row**  , and, in the text field, type the name of the new media type.
3. Click **Apply**.

1.2.5 Manage Search Query Integration

The Search Query Integration feature allows you create a custom search button on the **Advanced Search** screen. If a user clicks the custom search button, it launches a WebReport to which the user has permission. If the user has permission for multiple WebReports, clicking the custom search button will prompt the user to select a specific WebReport from the list of WebReports to which they have permission.

To make use of the Search Query Integration feature, you should also do the following:

- For one or more WebReports, on the **Properties** page, on the **Source** tab, set the data source type to **Search Query Launch**.
- For each WebReport with a Search Query Launch data source type, set the permissions so that only appropriate users can have access.

Technical Notes on Search Query Integration

If you are using multiple instances of Content Server, please note the following considerations:

- The enable option is server-specific and is stored in the `opentext.ini` file. You can use this file to restrict Search Query Integration to a specific Content Server instance.

  Conversely, if you require the same behavior for all servers in a cluster, you must enable the feature and set the button text for all servers in the cluster. You could achieve this by copying the same `opentext.ini` file to each server.
- On the **Manage WebReports Search Integration** page, the **WebReports With Data Source Set to Search Launcher** list is only visible when:
  - The Search Query Integration feature is enabled, and

  - you have set at least one WebReports' data source type to **Search Query Launch**.

- Activating or deactivating an individual WebReport takes effect system-wide. A disabled WebReports is unavailable on any Content Server instance.

To Enable Search Query Integration

To enable search query integration:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WebReports Search Integration**.
2. On the **Manage WebReports Search Integration** page, select the **Search Query Launch Button Enabled** check box.
3. Click **OK** to confirm that you will restart Content Server.

   Important
   You must restart Content Server to activate this feature.

4. In the **Search Query Launch Button Title** field, enter an appropriate name for the custom search button.
5. In the **Search Slice** field, enter a positive integer to set the number of results retrieved for each call of the search API.

   Changing this setting will impact the performance of WebReports that use search as a data source. In general, this setting should only be altered upon advice from OpenText Support. The default is “300”.
6. In the **Search Hard Limit** field, enter a positive integer to set the maximum limit on the number of results that any WebReport using search as a data source can return. The default is “50,000”.
7. Optional You can select **Search Debug Enabled** if you want extra debug information in the thread log when executing WebReports using search as a data source.
8. Optional In the **Additional Search Columns** section you can make optional columns available to WebReports that use search as a data source. The default columns available are “shortOTSummary” and “OTSummary”. Possible column values include: OTHotWords, parentRecord, and Score.

   In the **Search Column Name** field, enter the name of the column that will be returned by the search. In the **Output Name** field, enter the name of the column as it will appear in the output.
9. Click **Apply**.
10. Restart Content Server.

To Activate or Deactivate a Search Query Integration WebReport

To activate or deactivate a Search Query Integration WebReport:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WebReports Search Integration**.
2. On the **Manage WebReports Search Integration** page, in the **WebReports With Data Source Set to Search Launcher** list, on the row for the Search Query Integration WebReport that you want to activate or deactivate, do one of the following:
   - To activate the WebReport, select the **Active** check box.
   - To deactivate the WebReport, clear the **Active** check box.

   Note: The **WebReports With Data Source Set to Search Launcher** list is only visible when Search Integration is enabled and you have set at least one WebReports' data source type to **Search Query Launch**. For information about how to enable Search Integration, see "To Enable Search Query Integration" on page 12.
3. Click **Apply**.

1.2.6 Manage Tags and Sub-Tags

The **Manage Tags and Sub-Tags** page allows you to selectively show and hide enabled WebReports tags and sub-tags.

Notes

- You must restart Content Server after you make changes on this page.
- Disabled tags can no longer be added to a WebReport or an ActiveView template.
- Any existing reports that use a disabled tag or sub-tag will not process that tag or sub-tag.

To Manage Tags and Sub-Tags

To manage tags and sub-tags:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage Tags and Sub-Tags**.
2. Optional On the **Manage Tags and Sub-Tags** page, to switch the view between a list of all tags and sub-tags and a list of only the disabled tags and sub-tags, click **Toggle**.
3. Optional Scroll through the **Tag** list and the **Sub-Tag** list and, in the **Disable** column, select the check box next to any tag or sub-tag that you want to disable.

4. Click **Submit**.
5. Restart Content Server services to ensure that your changes take effect.

1.2.7 Manage Trusted Files

The **Manage Trusted Files** page allows you to configure a set of trusted external files that WebReports can use.

- **Trusted External Files for Data Sources**

  This is a whitelist of external file paths that are approved for use as a data source for a WebReport. At runtime, the resolved value of the file path stored on the **Source** tab of a WebReport is validated against this whitelist. If the requested source file path does not match any entries in this list, the WebReport will return an error message and will not execute.

- **Trusted File Paths for Server Export Destination**

  This is a whitelist of external files that are approved for use as an export destination for a WebReport. At runtime, the resolved value of the file path stored on the **Destination** tab of a WebReport is validated against this whitelist. The Server Export Destination feature in WebReports requires that the requested destination file path must be included in this whitelist. The WebReport will only execute if the requested destination file path matches an entry in this list. If the requested destination file path does not match an entry in this list, the WebReport will return an error message and will not execute.

Notes

- For security purposes, do not add "`C:*`" as a valid file path to either whitelist, as that would make system files available as data sources or as destinations.
- To use an External File as a data source or a destination of a WebReport, the file path must be included in the appropriate whitelist.

Consider the following criteria for valid file path entries in the whitelists:

- You can enter filename paths as a comma-separated list or as one path on each line.
- To minimize the number of entries required, use of the "*" wildcard character in the path.
- Matching is case-sensitive.
- Some examples of valid file paths include the following:
  - `C:\WRSourceData\mySourceFile.csv` – a path directly to the file.
  - `C:\WRSourceData\mySource*` – any file within `C:\WRSourceData` whose file name begins with "`mySource`" will be allowed.
  - `C:\WRSourceData\*` – any file path pointing to `C:\WRSourceData\` and its descendent directories will be allowed.

- `C:\WRSourceData*` – any file within `C:\` whose name begins with "`WRSourceData`", or whose file path begins with "`WRSourceData`" will be allowed. For example, `C:\WRSourceData\mySourceFile.csv` and `C:\WRSourceData22\myOtherFile.csv` would be allowed.

To Add an External File to the Trusted External Files for Data Sources List

To add an external file to the Trusted External Files for Data Sources list:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage Trusted Files**.
2. On the **Manage Trusted Files** page, in the **Trusted External Files for Data Sources** list, enter a filename path for the external file that you want to add.

   Important

   For security purposes, do not add "`C:*`" as a valid file path to this whitelist, as that would make system files available to be used as a data source.

   Notes

   - You can enter filename paths as a comma-separated list or as one path on each line.
   - To minimize the number of entries required, use of the "*" wildcard character in the path.
   - Matching is case-sensitive.
   - Some examples of valid file paths include the following:
     - `C:\WRSourceData\mySourceFile.csv` – a path directly to the file.
     - `C:\WRSourceData\mySource*` – any file within `C:\WRSourceData` whose file name begins with "`mySource`" will be allowed.
     - `C:\WRSourceData\*` – any file path pointing to `C:\WRSourceData\` and its descendent directories will be allowed.
     - `C:\WRSourceData*` – any file within `C:\` whose name begins with "`WRSourceData`", or whose file path begins with "`WRSourceData`" will be allowed. For example, `C:\WRSourceData\mySourceFile.csv` and `C:\WRSourceData22\myOtherFile.csv` would be allowed.
3. Click **Apply**.

To Add a Path to the Trusted File Paths for Server Export Destination List

To add a path to the Trusted File Paths for Server Export Destination list:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage Trusted Files**.
2. On the **Manage Trusted Files** page, in the **Trusted File Paths for Server Export Destination** list, add a trusted destination file path.

   Important

   For security purposes, do not add "`C:*`" as a valid file path to this whitelist, as that would make system files available to be used as a destination.

   Notes

   - You can enter filename paths as a comma-separated list or as one path on each line.
   - To minimize the number of entries required, use of the "*" wildcard character in the path.
   - Matching is case-sensitive.
   - Some examples of valid file paths include the following:
     - `C:\WROutputData\myOutputFile.csv` – a path directly to the file.
     - `C:\WROutputData\myOutput*` – any destination file within `C:\WROutputData\` whose file name begins with "`myOutput`" will be allowed to execute.
     - `C:\WROutputData\*` – any file path pointing to `C:\WROutputData\` and its descendent directories will be allowed as a destination path.
     - `C:\WROutputData*` – any file within `C:\` whose name begins with "`WROutputData`", or whose file path begins with "`WROutputData`" will be allowed. For example, `C:\WROutputData\myOutputFile.csv` and `C:\WROutputData22\myOtherFile.csv` would be allowed.
3. Click **Apply**.

1.2.8 Manage User/Group WR Triggers

Similar to "Manage WR Triggers " on page 26, the **User/Group WR Trigger** feature allows events occurring within Content Server and pertaining to users and groups to trigger a WebReport. To use this feature, you must select the **Enable User/Group WR Trigger** check box. The administration settings allow you to select different Content Server users or groups. For selected users or groups, you can select one or more trigger events by adding rows as needed. The **Inheritance** selection allows you to specify whether the scope applies to either Direct Members, or Direct *and* Indirect Members. When the selected trigger event occurs for any of the selected users or groups, the specified WebReport will run.

Important

The WebReports module is installed with Content Server, but is separately licensed. If the **Manage User/Group WR Trigger** link does not appear, please contact OpenText Support for information about how to purchase a license for the WebReports module.

If you have purchased and applied your WebReports license and you still cannot see the **Manage User/Group WR Trigger** link, please restart Content Server.

To Enable a User/Group WR Trigger

To enable a user/group WR trigger:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage User/Group WR Trigger**.
2. On the **Manage User/Group WR Trigger** page, select the **Enable User/Group WR Trigger** check box.
3. In the table, click the **here** link to add an individual User/Group WR event that will trigger a WebReport and perform the following steps:
   a. Do one of the following to apply the trigger to one or more specific users or groups:
      - In the **User/Group** column, click **Choose User or Group** to select the user or group.
      - In the **Global** column, select the check box to specify all users and groups.
   b. In the **Trigger Event** list, select the event that will trigger the WebReport. You have the option to click **Select All** to choose all the trigger events available.
   c. In the **Inheritance** list, select either **Direct Members** or **Direct and Indirect Members**.
   d. In the **WebReport Action** column, click **Browse Content Server** and select the WebReport to run for the current trigger condition.

e. Optional The **Add/Delete** column allows do one of the following:

- Click **Add Row** to add a new row for another trigger event.
- Click **Delete Row** to delete the current row and trigger event.

4. Click **Apply**.

1.2.9 Manage WebReports Conversion

Several of the WebReports destinations support document conversion. While it is common to convert a WebReport to a PDF-formatted report, WebReports supports any conversion provided by the external engine. The supported destinations include the following:

- E-mail
- Content Server Node
- Content Server Version
- Server
- Workflow (as an attachment)

If you want to use the external conversion feature, you must configure options on the **Manage WebReports Conversion** page of the **WebReports Administration** section of the **Content Server Administration** page.

The **Manage WebReports Conversion** page allows you to add, view, or modify information about the WebReports conversion directories. These directories are used in conjunction with a conversion tool, such as Adlib eXpress, to allow WebReports to perform exports in PDF or other formats.

To convert a WebReport, use the **Manage WebReports Conversion** page to define an input folder, an output folder, and the target format, and to set the conversion tool to monitor the input folder. Upon finding a file in the input folder, the tool should render the reportview in the defined format, and then deliver the format to the output folder and delete the input file.

Configuring the Conversion Agent

By default, the WebReports conversion agent is configured to run every 300 seconds, or five minutes. If you want the agent to run more frequently, you can configure it to run as often as every 60 seconds. You can set the frequency on the **Manage WebReports Conversion** page, in the **Converson Agent Sleep Interval** area.

Adlib eXpress Job Ticket Conversion

WebReports provides full support for Adlib eXpress Job Tickets. On the **Manage Conversion Directories** page, if conversion is enabled, you must specify the Input Directory and Output Directory.

If you want to enable XML Job Tickets, select the **Check this box if you would like to use XML Job Tickets...** check box, and then you have the option to provide detailed processing directives using an XML Job Ticket file. This XML file can be stored in Content Server with individual WebReports either sharing a single Job Ticket file or each providing their own processing instructions. For more information about XML Job Tickets, see the Adlib website (`https://www.adlibsoftware.com`).

If you enable the use of XML Job Tickets, you have the option to specify two additional conversion directories:

- Input Directory (when using XML Job Tickets), which should contain the WebReports intended for conversion.
- Output Directory (when using XML Job Tickets), which will contain the converted WebReports.

Notes

- For conversion to be enabled, you must specify the first set of Input and Output conversion directories. If you enable XML Job Tickets, you have the option to specify additional XML Job Ticket Input and Output directories. These are useful for servers running multiple instances of Content Server using different conversion methods. Current versions of Adlib eXpress support either conversion with Job Tickets or conversion without Job Tickets, not both simultaneously.
- If you are only using XML Job Ticket conversion, you must provide paths for the following fields:
  - Input Directory (when using XML Job Tickets)
  - Output Directory (when using XML Job Tickets)
- If you are using both XML Job Tickets and standard conversion, you must first specify the standard conversion directories and then specify the XML Job Ticket directories.
- The values entered on this page will be used to replace occurrences of `$DEFAULT_PATH` specified in the Job Ticket itself.
- For information about a basic example of a Job Ticket, see the Knowledge Center (https://knowledge.opentext.com/knowledge/cs.dll/Properties/62285007).

Technical Notes on External Conversion

The following factors that may affect the External conversion:

- WebReports uses the Content Server cache to determine the destination for each converted file. Since any cache has a finite life, it is possible, although unlikely, for converted files to be in the output folder that relates to an expired cache. For example, if you are exporting and converting on a system with disabled agents on all servers, the backlog of files will not be cleared unless there is a valid record in the current cache. In this case WebReports has lost the destination location and will not know where to send the files. For information about how to use the `collectConvDocCache.ini` setting to control the length of time before the cache expires, see "WebReports Preferences in the opentext.ini File" on page 41. Cache expiry is set to 120 minutes, by default, but can be increased as required. If the cache expires, the job cannot complete and you should delete or manually process the converted document as required.
- Content Server agents are used to look at the output folder and check for a converted file every five minutes. This means agents must be enabled for conversion to function.
- The amount of time it takes for a file to appear in the export destination may vary, depending on the system, from nearly instantaneous to 30 minutes. Influencing factors include the following:
  - When a file is written to the input folder
  - How often the conversion tool checks the input folder
  - How long to perform the conversion
  - When this hits in the cycle of the five-minute agent
- You can specify the folders at a network location as a UNC. For example: `\\192.168.255.20\disk1\PDF Conversion\input`
- In a Content Server clustered environment, you must define the directories as follows:
  - Define the input directory *on every instance* in the cluster because an export may come from any instance in a cluster.
  - Define the output folder *on the instance that is running the WebReports conversion agent* because that instance is the only one that monitors the output folder.

    Note: It is, however, good practice to specify the output folder on *all instances* of Content Server in the cluster.

To Configure an External Conversion Task

To configure an external conversion task:

1. Configure an external conversion engine, such as Adlib eXpress, to do the following:
   a. Watch an input folder for new files that need conversion.
   b. Store the converted documents in a specific output folder.
   c. Delete the source file from the input folder.
   d. Rather than appending the new file extension onto the old one, replace the source type file extension, for example `.html`, `.doc` or `.xls`, with the destination file extension, for example `.pdf`.
2. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WebReports Conversion**.
3. On the **Manage WebReports Conversion** page, in the **Conversion Agent Sleep Interval** box, enter a value (in seconds) for the interval the conversion engine should wait before checking the input folder.

   Notes

   - This interval contributes to the amount of time it takes for the converted file to appear in the destination folder.
   - The default setting is 300 seconds, or five minutes.
4. In the **Input Directory** box, specify the path of the input folder.
5. In the **Output Directory** box, specify the path of the output folder.

   Important

   The operating system account that runs the Content Server service must have permission to write a file to the input folder and to delete a file from the output folder.
6. Optional To configure Adlib eXpress Job Ticket conversion, select the **Check this box if you would like to use XML Job Tickets...** check box and provide the following mandatory directories:
   a. In the **Input Directory (when using XML Job Tickets)** field, enter the XML Job Ticket input directory to use when files are converted using XML Job Tickets.
   b. In the **Output Directory (when using XML Job Tickets)** field, enter the XML Job Ticket output directory to use when files are converted using XML Job Tickets.
7. Click **Apply**.
8. Restart Content Server.

1.2.10 WebReports Schedules

On the **Destination** tab of the **Properties** page for a WebReport, developers, or users with *reserve* permission, can set that WebReport to run automatically at scheduled times, or for a fixed number of times, or forever, starting from a specific date and time. The interval between repeated WebReport runs can range from five minutes to 520 weeks. Each user can set one schedule per WebReport. Scheduled WebReports run in the background and output their results to a Content Server document node or to an email.

The WebReports scheduling agent schedules the execution of WebReports nodes. On the **WebReports Administration** > **Manage WebReports Schedules** page, you can set the scheduling agent sleep interval. By default, the WebReports **Scheduling Agent Sleep Interval** is set to run every 300 seconds, or five minutes. If you want to execute a WebReport more frequently, the interval can be shortened to a minimum of 60 seconds.

Notes

- Multiple users can use the scheduling agent to set multiple schedules on the same WebReport.
- After setting new values for an agent with a new values, you must restart Content Server.

If any Scheduling Agent Sleep Interval is less than 300 seconds, the **Destination** tab of the **Properties** page for that WebReport allows you to manually enter a **Minutes** value for the scheduled report. This allows for more granular control of how often the WebReport will run.

For example, to configure a WebReport to run every two minutes, set the **Scheduling Agent Sleep Interval** to "120" seconds. Save the change and restart Content Server. On the **Destination** tab of the **Properties** page for that WebReport, click **Enter Minutes (0-59)** . Now that the **Minutes** field has become a text field, enter the value of "2" to have the WebReport execute every two minutes.

Configuring Change Agents for WebReports Scheduling

Although Content Server notifications are optional for WebReports scheduling, you must enable Change Agents on your Content Server instance. By default, Change Agents are enabled in the `[options]` section of the `opentext.ini` file for at least one instance of Content Server will contain `EnableAgents=TRUE`.

If you are unsure about how to configure **Change Agents** for your particular Content Server instance, please contact OpenText Support.

Database Schema

Installation or upgrade of the WebReports module, creates the following Content Server database tables:

- The `WEBREPORTS` table stores WebReports schedule information. This table has a two column key: `USERID` and `NODEID`, where `NODEID` is the nodeid of the WebReport.
- An additional table called `WEBREPORTSTATS` is created when the module is installed or upgraded.

To Configure the Scheduling Agent

To configure the scheduling agent:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WebReports Schedules**.
2. On the **Manage WebReports Schedules** page, in the **Scheduling Agent Sleep Interval** box, enter the amount of time (in seconds) that the agent should wait between executions.
3. Click **Save Agent Changes**.
4. Restart Content Server.
5. Optional If you have set the **Sleep Interval** to a value less than 300, you need to access the properties for the WebReport and do the following:
   a. Click the **Destination** tab.
   b. In the **Output Destination** list, select one of the following destinations that support scheduling:
      - Content Server
      - E-mail
      - Server
      - Workflow
      - Form
      - FTP
   c. Select the **Set Schedule** check box.
   d. In the **Every** area, click **Enter Minutes (0-59)**.
   e. In the **Minutes** box, enter the amount of time (in minutes) that defines the repeat interval on which WebReports will run. For example, if you entered "120" in the **Scheduling Agent Sleep Interval** field, enter "2" in the **Minutes** field to have the WebReport execute every two minutes.
   f. Click **Update**.

To Manage Schedules

To manage schedules:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WebReports Schedules**.

2. Optional On the **Manage WebReports Schedules** page, each row in the table lists a schedule set by one user for one WebReport. For each schedule, you can do one or more of the following:

   - Disable the schedule by clearing the **Enabled** check box. This will retain the schedule information, but the report will not run until a user or administrator re-enables the report.
   - Enable or re-enable a disabled schedule by selecting the **Enabled** check box.
   - Delete a schedule by selecting the **Delete Schedule** check box.

   Note: Content Server automatically deletes a schedule if the user that created the schedule is deleted or if the WebReport itself is deleted.

3. Click **Apply**.

1.2.11 Manage WebReports Scripting

The **WebReports Scripting** page allows you to review and approve all WebReports that contain Oscript sections within their reportviews. By default, Oscript is disabled for a WebReport until a system administrator enables it. An administrator can also enable Oscript by adding a new version to the reportview. The approving system administrator is responsible for ensuring that reports are safe and avoid issues such as infinite loops.

Oscript in WebReports is heavily restricted to make it secure. For example there is no access to global variables or many of the Builder Packages. You can change which features are allowed or restricted by modifying settings in the `opentext.ini` file.

OpenText recommends that all reports containing Oscript are developed and tested on a development instance of Content Server prior to being added to a production instance of Content Server.

Each Oscript report can be enabled or disabled individually by selecting or clearing the check box next to a report. You can also enable or disable the use of Oscript for an individual WebReport from that WebReport's **Properties** page by clicking the **Specific** tab and then selecting or clearing the **Oscript Scripting Enabled** check box.

Note: WebReports that include disabled Oscript sections can still run, but all Oscript calls will be ignored and not processed.

To Manage WebReports Scripting

To manage WebReports scripting:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WebReports Scripting**.
2. Optional On the **Manage WebReports Scripting** page, do one of the following:
   - Enable a disabled Oscript section by selecting the **Oscript Enabled** check box for that Oscript section.
   - Disable a previously-enabled Oscript section, by clearing the check box for that Oscript section.

   Tip: You can also enable or disable Oscript for an individual WebReport from that WebReport's **Properties** page by selecting the **Specific** tab.
3. Click **Apply**.

1.2.12 Managing WebReports Services

The **Manage WebReports Services** page allows you to manage access to WebReports Services by the following restriction levels:

- Disable the WebReports Services feature.
- Restrict a tag or sub-tag from use by WebReports Services.
- Require a secure token for a tag or sub-tag with special security considerations for use by WebReports Services.

Important

The WebReports module is installed with Content Server, but is separately licensed. If the **Manage WebReports Services** link does not appear, please contact OpenText Support for information about how to purchase a license for the WebReports module. If you have purchased and applied your WebReports license and you still cannot see the **Manage WebReports Services** link, please restart Content Server.

For more information about WebReports Services, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

To Enable or Disable a WebReports Service

To enable or disable a WebReports service:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WebReports Services**.
2. On the **Manage WebReports Services** page, configure one or more of the following options:
   - **WebReport Services**

     Select the **Disable WebReport Services Feature** check box to disable WebReports services and clear the check box to enable it.
   - **Restrict the use of a tag or sub-tag by WebReports Services**

     On the **Restricted Services** list, click or **CTRL** + click to select one or more tags or sub-tags that you would like to restrict from use by WebReports Services.

     Note: An attempt to use a restricted tag or sub-tag will trigger an error message.
   - **Require a secure token for the use of a tag or sub-tag**

     On the **Secure Services** list, click or **CTRL** + click to select one or more tags or sub-tags which will require a secure token for use by WebReports Services.

     Note: An attempt to use a secure services tag or sub-tag without providing a secure token will trigger an error message.
3. Click **Apply**.

1.2.13 Manage WR Triggers

The **WR Triggers** feature allows events occurring within Content Server to trigger a WebReport. Using the **Manage WR Triggers** page, you can enable or disable different Content Server node sub-types. If you enable**WR Triggers** for a node sub-type, the **WR Trigger** Properties tab will appear for all nodes of that sub-type. If you disable **WR Triggers** for a node sub-type, no code associated with WebReports will execute for that node subtype and the **WR Triggers** option will not appear in the function menus for that node. If you disable all node sub-types, **WR Triggers** has no performance cost associated with it. If you install or upgrade to a version of WebReports that includes WR Triggers, all node sub-type are disabled by default.

Important

The WebReports module is installed with Content Server, but is separately licensed. If the **Manage WR Triggers** link does not appear, please contact OpenText Support for information about how to purchase a license for the WebReports module. If you have purchased and applied your WebReports

license and you still cannot see the **Manage WR Triggers** link, please restart Content Server.

To Manage WR Triggers

To manage WR triggers:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Manage WR Triggers**.
2. On the **Manage WR Triggers** page, in the **Node Subtypes** list, click the node type to which you want to apply the WR Trigger.
3. Click **Apply**.
4. Restart Content Server.

1.2.14 Miscellaneous WebReports Settings

The **Miscellaneous WebReports Settings** page allows you to set the values of miscellaneous keys in the WebReports section of the `opentext.ini` file.

Currently the only key that can be set is `EmailAddressMaxCharacters`. This key sets the maximum number of characters that can be used for the email address string of a WebReport with a destination of email.

If an attempt is made to change the value to a non-integer value then the default value of 10000 characters will be used.

Important

The WebReports module is installed with Content Server, but is separately licensed. If the **Miscellaneous WebReports Settings** link does not appear, please contact OpenText Support for information about how to purchase a license for the WebReports module. If you have purchased and applied your WebReports license and you still cannot see the **Miscellaneous WebReports Settings** link, please restart Content Server.

To Manage Miscellaneous WebReports Settings

To manage miscellaneous WebReports settings:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **Miscellaneous WebReports Settings**.
2. Optional On the **Miscellaneous WebReports Settings** page, in the **Set the maximum number of characters** box, enter the a positive integer value for the maximum number of characters for a destination email address.
3. Click **Apply**.

1.2.15 License Key Functions

You must install a license key before being able to run or export a WebReport.

Note: The WebReports module is installed with Content Server, but is separately licensed.

A new license key is required for WebReports with each major version release of Content Server. For example, upgrading from Content Server 10.5.0 to 16.0.0 requires a new license. No new license is required when applying cumulative Content Server updates.

A new license key is not required for each minor version of the module. For example, upgrading from WebReports 10.0.0 to WebReports 10.0.1 does not require a new key.

Viewing License Usage Reports

WebReports provides a licensing status report. This report indicates when license usage has exceeded 80% of available WebReports licenses. It also indicates when the number of licenses has been exceeded. This usage information makes it simple to place orders for additional WebReports user licenses when required. It is advisable to review the report on a regular basis to ensure that there are sufficient user licenses available.

Note: When a Content Server user that has used WebReports is deleted, the license for that user is released automatically and becomes available for use by another user.

WebReports also provides warnings to report developers if the number of users exceeds the number of licenses by 10% or more. These warnings will appear when editing or performing other configuration actions on a WebReport but do not affect the running of reports.

To View the License Usage Report

To view the license usage report:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **WebReports Licensing**.
2. On the **WebReports Licensing** page, click **License Management**.
3. On the **Manage Licenses** page, in the **Options** menu, click **License Report**.

Licensing WebReports

The WebReports module is installed with Content Server, but is separately licensed. If the **Add Item** menu does not include WebReport, you may need to contact OpenText Support for information about how to purchase a license for the WebReports module.

To Enter the WebReports License Key

To enter the license key:

1. On the **Content Server Administration** page, in the **WebReports Administration** section, click **WebReports Licensing**.
2. On the **WebReports Licensing** page, click **License Management**.
3. On the **Manage Licenses** page, in the **Options** menu, click the **License Management** tab.
4. On the **License Management** page, click **Manage Licenses**.
5. OpenText Directory Services will open in a new tab. Sign in to Directory Services to apply your WebReports license. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS-IWC)*.
6. After Directory Services confirms the WebReports license is okay, return to Content Server and restart the Content Server admin services.

1.2.16 WebReports Node Administration

Important

The WebReports module is installed with Content Server, but is separately licensed. If the **WebReports Node Administration** link does not appear, please contact OpenText Support for information about how to purchase a license for the WebReports module. If you have purchased and applied your WebReports license and you still cannot see the **WebReports Node Administration** link, please restart Content Server.

The **WebReports Node Administration** page identifies any existing WebReports nodes or ActiveView nodes that use syntax that is out-of-date for the current release. Any WebReports nodes or ActiveView nodes that continue to use out-of-date syntax may not work as they did in previous releases. To maintain functionality, use this utility to update the selected nodes with the current syntax. The utility adds a version containing the updated syntax to the selected node. If the post-update result is not as you expect, you can roll back the node using the **Version** tab in WebReports or ActiveView.

1.2.17 WebReports Sub-Tag Builder

The **WebReports Sub-tag Builder** page shows a list of custom sub-tags that have been added to the system by being "dropped" into the subtags folder in the WebReports installation directory. New or replacement sub-tags only take effect after you execute the **Build All Sub-tags** option and restart Content Server.

Important

The WebReports module is installed with Content Server, but is separately licensed. If the **WebReports Sub-tag Builder** link does not appear, please contact OpenText Support for information about how to purchase a license for the WebReports module. If you have purchased and applied your WebReports license and you still cannot see the **WebReports Sub-tag Builder** link, please restart Content Server.

Error messages will appear on the **WebReports Sub-tag Builder** page if one or more sub-tags in the WebReports Ospace, or one or more sub-tags in the `subtags` folder, are invalid and do not compile.

If your installation of WebReports has not been updated since loading the module and you still see error messages on the **WebReports Sub-tag Builder** page, please contact OpenText Support and inform them of these errors.

For information about custom sub-tags, see "Custom Sub-tags" on page 30.

1.2.18 Custom Sub-tags

Use ordinary text documents to create your own sub-tags and place them into a folder on the server. This allows you to generate sub-tag features customized for your needs that you can include in your WebReport reportview and use just like any other sub-tag. There are no restrictions on what you can code in a custom sub-tag because they are located on the server. This means that OScript packages that are normally blocked for use in a WebReport reportview with server-side scripting are available in a custom sub-tag. This results in a WebReports sub-tag that behaves just like any other sub-tag that is packaged with WebReports. A sample custom sub-tag called `sample.txt` is located in the *<Content Server_home>*`\module\webreports_<n>_<n>_<n>\subtags` folder on the server where you installed Content Server. Use this file as a template to build new sub-tags and save them into the same folder. In addition, there is an associated *<Content Server_home>*`\module\webreports_`*<n>*`_`*<n>*`_`*<n>*`\subtags\sample.txt.json` file that provides configuration for the sub-tag. This `sample.txt.json` file is required for sub-tag functionality.

Notes

- To make the new custom sub-tags available, you must restart the Content Server service.
- To test that your sub-tag code compiles correctly, use the **Content Server Administration** > **WebReports Administration** > **Build All Sub-tags** page. For more information, see "WebReports Sub-Tag Builder" on page 30.

Example 1-1: Creating a custom sub-tag called ADDINTEGERS which adds together the integers passed into the tag

1. Create a copy of the `sample.txt` file and rename it to `addintegers.txt`.
2. Create a copy of the `sample.txt.json` file and rename it to `addintegers.txt.json`.

3. Replace the text in the `addintegers.txt` file with the following:

```
function Object Execute( \
    List args = {} )

    // Calling the "dataAs..." casting functions will
operate on the data passed to the sub-tag from any
previous tags (.fData), provided the functions are not
passed a specific value to cast.
    // For the [LL_REPTAG_'2' ADDINTEGERS:3:4 /] example
this is operating on the tag literal integer 2 from
[LL_REPTAG_'2'

    Integer value1 = .dataAsInteger()

    // Each tag argument separated by a colon is added to
a list.
    // For the [LL_REPTAG_'2' ADDINTEGERS:3:4 /] example
the first argument, 3 would be accessible using args[1] as
it is the first element in the args list
    // The second argument, 4 would be accessible using
args[2] as it is the second element in the args list

    Integer value2 = .dataAsInteger( args[1] )
    Integer value3 = .dataAsInteger( args[2] )

    if ( IsDefined( value1 ) && IsDefined( value2 ) )

        .fData = value1 + value2

        if ( IsDefined( value3 ) )

            .fData += value3

        end

    else

        return .SetError( 'ADDINTEGERS', 'Parameters must
be valid integers', 1001 )

    end

    return this
end
```

This will create a custom sub-tag called `ADDINTEGERS`. The sub-tag takes between one and two parameters and adds them to the value in the main tag, returning the total. The code contains error handling, so if any of the parameters are not integers, an error message will be returned instead of the total.

4. Save the updated `addintegers.txt` file.

5. Replace the text in the `addintegers.txt.json` file with the following:

```
{
"fMinParms":1,
"fMaxParms":2,
"fShowInTagGuide":true,
"OverrideRule": "UPVERSION",
"Version": "1.0.1",
"Documentation":"<p>
    Add the value of the main tag to the parameters passed
in.
</p>
<br>
Examples:<br><br>

    [LL_REPTAG_'2' ADDINTEGERS:3 /] returns 5.<br><br>

    [LL_REPTAG_'2' ADDINTEGERS:3:4 /] returns 9.<br><br>

    [LL_REPTAG_'x' ADDINTEGERS:'y' /] returns '*
ADDINTEGERS: Parameters must be valid integers *'.<br><br>
"
}
```

The keys in the JSON file are populated as follows:

- `fMinParms` is an integer defining the minimum number of parameters that the sub-tag will accept. If less than the minimum parameters are passed in, then an error will be returned.
- `fMaxParms` is an integer defining the maximum number of parameters that the sub-tag will accept. If more than the maximum parameters are passed in, then an error will be returned.
- `fShowInTagGuide` defines whether the sub-tag will appear in the online Tag Guide. Can be set to `TRUE` or `FALSE`.
- `OverrideRule` defines the circumstances where the sub-tag will be applied. It can have three values:
  - `UPVERSION`: the custom sub-tag will only be added if it doesn't already exist, or if there is already a version of the sub-tag in your version of WebReports, then the custom sub-tag will take precedence over the existing sub-tag only if the custom sub-tag has a higher version number.
  - `NEWONLY`: the custom sub-tag will only be added if it doesn't already exist in your version of WebReports.
  - `ALWAYS`: the custom sub-tag will always be applied, regardless of whether or not it already exists in your version of WebReports.

Notes

- If a version field is present, the default is: `UPVERSION` , otherwise `ALWAYS`.
- Documentation defines the text that will be displayed in the online Tag Guide if `fShowInTagGuide` is set to `TRUE`.

6. Restart Content Server.

Example 1-2: Call the new custom sub-tag

Call the new custom sub-tag from any WebReport reportview as follows:

- `[LL_REPTAG_'2' ADDINTEGERS:3 /]` returns 5.
- `[LL_REPTAG_'2' ADDINTEGERS:3:4 /]` returns 9.
- `[LL_REPTAG_'x' ADDINTEGERS:'y' /]` returns "`* ADDINTEGERS: Parameters must be valid integers *`".

1.3 Content Server Functions Available for WebReports

In addition to the unique functions provided on the **WebReports Administration** page, basic Content Server document management functions are provided to allow full version control and other features for the reportviews.

This means that WebReports are subject to all of the administrative options that would normally be available for Content Server documents. In particular, any document administrative options, such as **Administer Item Control**, will also apply to reportview files. For more information about these other Content Server functions, see the following topics:

- "Enabling Content Server Functions for WebReports" on page 35
- "Setting Permissions for WebReports Users" on page 39
- "WebReports Preferences in the opentext.ini File" on page 41
- "Accessing Content Server Logs" on page 47

Reportview files are fully indexed, allowing them to be located via the search facility, though most users will not have read permissions. For more information about reportviews, see the following topics:

- "Cached Views" on page 38
- "Default Reportviews" on page 38

1.3.1 Enabling Content Server Functions for WebReports

You can enable Content Server functions for WebReports through the main Content Server administration pages.

- You can restrict the creation of WebReports to a selected user or group.
- You can enable the standard Content Server events for WebReports.
- You can store WebReports in the Content Server Report Volume.
- You can export WebReports to email.

Restricting the Ability to Create New WebReports

Like most other Content Server objects, you can control WebReport objects using the Content Server "object factory" to restrict the creation of WebReports to a selected group of users.

To Restrict the Ability to Create New WebReports

To restrict the ability to create new WebReports:

1. On the **Content Server Administration** page, in the **System Administration** section, click **Administer Object and Usage Privileges**.
2. On the **Administer Object and Usage Privileges** page, in the **Object Privileges** table, scroll down until you find **WebReport** under the **Object Type** heading.

   Note: By default, the **Create Object Status** for **WebReport** is *Restricted*, and the action available in the **Actions** column is **Restrict**.

3. Optional To restrict the ability to add a new **WebReport** by accessing it from the **Add Item** menu, do one of the following:

   - If the **Create Object Status** is *Unrestricted*, in the **Actions** column, click **Restrict**.
   - If the **Create Object Status** is *Restricted*, in the **Actions** column, click **Edit Restrictions**.

4. On the **Edit Group: WebReports** page, in the **To add members** pane, use the **Find** lists to search for the person or group to whom you want to restrict access.
5. In the search results table, on the row for the matching name of the person or group, in the **Actions** column, select the **Add to group** check box, and then click **Submit**.
6. In the **Modify Group** pane on the left, verify that the name of the person or group has been added to the **Current Group Members** list and then click **Done**.

Administering WebReports Audit Events

Since the reportview for a WebReport is like any other Content Server document, you can enable and use all of the standard Content Server events for a WebReport.

Some WebReports auditing events are enabled by default. To enable those auditing events that are disabled by default, use the Content Server **Set Auditing Interests** page.

To write the `RUN` or `EXPORT` events for a WebReport to the `AUDIT` log, you must enable the **Run/Export Audits Enabled** option on the **Specific** tab of the **Properties** page for each WebReport. This is disabled by default to provide granular control to stop the audit logs from overflowing.

To Enable Audit Events for a WebReport

To enable audit events for a WebReport:

1. On the **Content Server Administration** page, in the **System Administration** section, click **Administer Event Auditing**.
2. On the **Administer Event Auditing** page, click **Set Auditing Interests**.
3. On the **Set Auditing Interests** page, in the **Events** area, select the **WebReports Run** check box and then click **Set Interests**.
4. Optional For each WebReport, to run `EXPORT` and `RUN` audit events, which are disabled for WebReports by default, do the following:
   a. From the **Functions** menu for the WebReport, select **Properties** > **Specific**.
   b. On the **Specific** tab, select the **Run/Export Audits Enabled** check box.
   c. Click **Update**.

Storing WebReports in the Report Volume

The WebReport object has been set up to behave like a LiveReport for any system functions that are associated with reports. In particular, you can save a WebReport in the Reports Volume. This can be viewed using the **LiveReports** tab, found on the **Reports** page, which you can access from the **Personal** menu.

To Store a WebReport in the Report Volume

To store a WebReport in the Report volume:

1. When adding, moving, or copying a WebReport, on the **Add**, Copy, or Move page, in the **Create In** area, click **Browse Content Server**.

   - If you are creating a new WebReport, on the **Add** page, in the **Create In** area, click **Browse Content Server**.
   - If you are moving an existing WebReport, on the **<***WebReportName***>** page, in the **Move to** area, click **Browse Content Server**.
   - If you are copying an existing WebReport, on the **<***WebReportName***>** page, in the **Copy to** area, click **Browse Content Server**.

   When saving your WebReport, during an add, move or copy operation, in the **Create In** box, click the select link next to **Content Server**.

2. On the**Select Container to Create In** page, scroll to **Content Server Reports** and click the **Select** link.

3. Optional To see the WebReports that you have saved to the Reports Volume, from the Content Server toolbar, do one of the following:

   - Click **Personal** > **Reports** and then click the **LiveReports** tab.
   - Click **Project** > **Reports** and then click the **LiveReports** tab.

Exporting to E-Mail

WebReports can export to multiple locations, including e-mail. For e-mail exports to work correctly, you must provide the following details on the **Configure Notifications** page in the Content Server Administration:

- SMTP Server ID
- SMTP Port
- Content Server Host Name

Note: Although you must provide these details, you do not need to enable Content Server Notifications.

To Enable Exporting to E-Mail

To enable exporting to e-mail:

1. On the **Content Server Administration** page, in the **Notification Administration** section, click **Configure Notification**.
2. On the **Configure Notification** page, in the **SMTP Settings** area, do the following:
   a. In the **SMTP Server ID** box, enter the name of the SMTP server. The default for most SMTP servers is "mail".
   b. In the **SMTP Port** box, enter the port on which the SMTP server listens. The default for most SMTP servers is "25".
   c. In the **Content Server Host Name** box, enter the fully qualified DNS name of the primary Content Server host. For example, "contentserverhost.mycorp.com".

   Note: Although you must provide these details, you do not need to enable Content Server Notifications.
3. Click **Submit**.

1.3.2 Cached Views

The first time that you run a WebReport, a cached file is created for the current version of the reportview file. This functionality is similar to that used for WebForms. These cached views can be found at: *<Content Server_home>*`/temp/WRviews`.

Because it is sometimes desirable for users to execute multiple versions of the same WebReport, all versions of cached files are maintained in this folder. You can delete the contents of this folder at any time. These files are created as soon as any user runs a WebReport for the first time.

1.3.3 Default Reportviews

When you create a WebReport, you can either browse for a reportview on your desktop or choose from a list of standard reportview templates in the **Use a Default Reportview** list. These templates are stored as text files in Content Server in the module directory for WebReports. For example, *<Content Server_home>*`\module\webreports_x_x_x\defaultreportviews\10_basic_report.txt`

As administrator, you can modify the template list by editing or renaming existing files, or by adding additional files to the `defaultreportviews` folder. Files in the `defaultreportviews` folder will be selectable in the **Use a Default Reportview** list.

Note: Default reportview filenames should use the `.txt` extension and should start with *<x>*_ where *<x>* is a number that determines the order of the default reportviews in the menu.

You might want to adapt or adding new default reportviews for the following reasons:

- To include standard corporate branding to reports.
- To provide specific export reportviews for exporting Content Server data to other applications. For example, CSV or WordML.

For more information about the default reportviews provided with WebReports, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

Sample Reportviews

The WebReports module is packaged with various sample reportviews, which are installed at: `<Content Server_home>\module\webreports_x_x_x\examplereportviews`

Since all WebReports developers will find these reportviews useful, you may want to consider moving these reportviews to a standard location that is accessible to the appropriate communities of users. For example, you could create an area in Content Server called "WebReport development" that contains the sample WebReports, which you can build using the default sample reportviews. Alternatively, you can upload the reportview files to Content Server for developers to retrieve.

1.3.4 Setting Permissions for WebReports Users

Since running a WebReport combines aspects of running a report as well as managing a document, there are some unique considerations for setting WebReports permissions.

"WebReports Users Common Roles and Permission Settings" on page 39 indicates the most common roles for WebReports users and their recommended permissions settings.

Table 1-1: WebReports Users Common Roles and Permission Settings

WebReports User Role	Minimum Permissions	Available Commands
General users who are not allowed to run or edit the WebReport.	*See*	Make Shortcut

WebReports User Role	Minimum Permissions	Available Commands
End Users allowed to run the WebReport. This will be the main permission setting for all End Users of WebReports. It differs from normal documents where users would need to View, Fetch, or Download. With WebReports these commands are only used for the users who actually create or edit the reportviews.	*See Contents*	Open, Export, and standard See Contents commands. Examples of standard See Contents commands include Set Notification and Add to Favorites
WebReport Developers who might create or edit other WebReports but who are not allowed to or required to have access to modify this specific WebReport. This enables them to take the reportview, for example using download, and reuse it in another WebReport for which they have *Reserve* permission. Note that this may appear counter-intuitive.	*Modify*	Copy, Open, Download, View
WebReport Developers who are responsible for modifying the reportview, selecting different data sources and or creating scheduled instances of a report. These are typically the WebReport owners; however, they do not necessarily need to have *Delete* or *Edit Permissions* access.	*Reserve*	Add Reportview Version, Edit Reportview, Properties: Specific
Administrative Users	As Required. Normal permissions behavior.	Examples include Delete and Permissions.

In addition to permissions for the WebReport itself, any user running the WebReport must also have *See Contents* permission for the associated data source, which is specified when the WebReport is first created or later using the **Properties: Source** page.

Developers with *Reserve* permission for a WebReport do not necessarily require permission to edit or view the contents of the associated data source, but they will have permission to change which data source the WebReport object is associated with.

1.3.5 WebReports Preferences in the opentext.ini File

You can set a number of WebReports configuration preferences by editing the `opentext.ini` file. "WebReports Configuration Options in the opentext.ini File" on page 41 describes the configuration options.

Notes

- Be cautious about making changes to settings in the `opentext.ini` file.
- After you make changes, you must restart Content Server.
- If you are unsure about a change, make sure to consult OpenText Support.

Table 1-2: WebReports Configuration Options in the opentext.ini File

Field Name	Type	Description
addsearchbutton	String or null	Describes the label used for the button to launch a WebReport from Content Server search. This should be set from the **Content Server Administration** page. For more information about addsearchbutton, see the *OpenText Content Server - WebReports (LLESWEBR-UGD)*.
searchslice	Integer	Do not alter this setting in the INI file directly. To change this setting, see "To Enable Search Query Integration" on page 12. Determines how many results are retrieved for each call of the search API. Changing this setting will impact the performance of WebReports that use search as a data source. In general, this setting should only be altered upon advice from OpenText Support. Default = 300

Field Name	Type	Description
searchHardLimit	Integer	Do not alter this setting in the INI file directly. To change this setting, see "To Enable Search Query Integration" on page 12. Determines a maximum limit on the number of results that a WebReport using search as a data source can return. Default = 50,000
searchDebug	true / false	Do not alter this setting in the INI file directly. To change this setting, see "To Enable Search Query Integration" on page 12. Turns on extra debug information in the thread log for executing WebReports with search as a data source. Default = False
EnableOscriptReportviews	true / false	Controls whether Oscript is permitted within reportviews. Default is true. When set to false no oscript within any reportview will be compiled or executed. This turns the scripting feature off completely on the Content Server. Default = True For more information about EnableOscriptReportviews, see the *OpenText Content Server - WebReports (LLESWEBR-UGD)*.
OscriptAllowFunction1	List of strings	A list of individual oscript package functions to be permitted within a reportview. For more information about addsearchbutton, see the *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

Field Name	Type	Description
OscriptAllowWholePkg	List of strings	A list of oscript packages to be permitted within a reportview. For more information about OscriptAllowWholePkg, see the *OpenText Content Server - WebReports (LLESWEBR-UGD)*.
MaxNestedSubWebReports	Integer	Controls the upper limit of how many levels of sub-WebReports may be nested. If this field is empty, default = 5
VersionInput	String dir path	Specifies the input directory for a converter (e.g. AdLib eXpress). This should be changed via the **Manage WebReports Conversion** administration page.
ConversionDir	String dir path	Specifies the output directory for a converter (e.g. AdLib eXpress). This should be changed using the **Manage WebReports Conversion** administration page.
additionalSearchColumns	List of strings or list of lists	Determines the optional columns available to WebReports that use search data sources. Possible values include: OTHotWords, OTSummary, parentRecord, Score, shortOTSummary. These can be specified as strings within a list or as a list of two element lists where the first element is the column coming back from search and the second column is what you want it to be called in the output. It should be noted that these values are made available via the search index. If the underlying structure of the index were to change in future releases, these columns may no longer be available.

Field Name	Type	Description
AllowFormCustomview	true / false	Control whether WebReports can be added to form templates as views. This feature is visible by navigating to the views tab of a form template. The options **Add WR Power View** will be present if the feature is enabled. The feature is enabled by default.
WRNodeMIMETypes	List of strings	Determines the list of MIME types that a WebReport node itself may be set to. Effectively, this is the MIME type of the reportview. Typically, WebReports are assigned the `text/plain` MIME type, but sometimes it may be useful to permit `text/html` or other MIME types.
destinationmimetypes	List of strings	The is the MIME type of the output that will be created by WebReports. Depending on client settings, this can determine which application is used to open the resulting document.
WRTriggerSubtypes	List of integers	Controls the sub-types that are able to initiate WebReports from the WR Trigger feature. All sub-types are disabled by default.
DisallowServices	true / false	Enables or disables the WR Services feature. WR Services is enabled by default.
livelinkcategory_maxNumCats	Integer	Defines the maximum number of categories selectable via the category as a data source feature.

Field Name	Type	Description
livelinkcategory_NumAttrAcross	Integer	Defines the maximum number of attributes that should be displayed on a single line from the category as a data source feature.
livelinkcategory_attrDelimiter	String	This character is used to separate multivalue attributes in the output from the category as a data source feature.
livelinkcategory_maxNumRows	Integer	Controls the maximum number of results that are retrievable from the category as a data source feature.
AllowFormCustomview	true / false	Enables or disables the WR Power View feature which allows a WebReport to be used as a form view.
collectConvDocCache	Integer	For WebReports that use certain destinations that allow for conversion of the WebReport output before delivery to its final destination, the document is stored in a cache which is cleared by an agent after 120 minutes by default. This setting specifies an alternate expiry time for the cache.

Field Name	Type	Description
SecureServicesSubtags	List of Strings	This setting is used to define any sub-tags that require special security considerations when being used through the WR Services feature. If any of the specified sub-tags are included in a WR Services request then the software will expect that a secure token has also been provided in the request. For example, `?func=webreports.runservice&servicetype=gettagdata&tagdata=1000&subtags=useraction:delete` would create an error message unless a valid security token had previously been requested (`&servicetype=gettoken`) and added to the request like this: `?func=webreports.runservice&servicetype=gettagdata&tagdata=1000&subtags=useraction:delete&securerequesttoken=123456....` This setting is automatically added with a default list on new, full installs but if the software is installed as an Update the default setting is added as a comment. The current set of sub-tags included by default are: SecureServicesSubtags={'CATACTION','NODEACTION','SCACTION','PERMACTION','RMACTION', 'USERACTION','WFACTION','USERINFO','RMINFO','NODEINFO','EMAILINFO','WFINFO','AUDITINFO',

Field Name	Type	Description
		'VERINFO','SCINFO','POINFO'}
UpdateNumber	Integer	The software uses this setting to manage software update versions. You should only change this setting with guidance from OpenText Support.

Example 1-3: The following example illustrates a typical arrangement:

```
[WebReports]
addsearchbutton=Custom Report
VersionInput=C:\AdLib eXpress\Input
ConversionDir=C:\AdLib eXpress\Output
OscriptAllowFunction1={'Scheduler.debugbreak','Web.CRLF','Web.
Escape','Web.Unescape','Web.DecodeForURL','Web.EncodeForURL','
Web.Format','Web.EscapeHTML','Web.EscapeXML'}
OscriptAllowWholePkg={'Assoc','Bytes','Date','List','Math','Pa
ttern','RecArray','Str','String','Boolean','undefined','void',
'integer','real','record'}
EnableOscriptReportviews=true
MaxNestedSubWebReports=10
additionalSearchColumns={{'OTSummary','Summary'},
{'shortOTSummary','shortSummary'},OTHotWords,Score,parentRecor
d}
```

1.3.6 Accessing Content Server Logs

WebReports writes to the Content Server Thread Logs. Thread logging is disabled by default in Content Server and must be enabled. For more information, see *OpenText Content Server - Administering Content Server (LLESWBA-AGD)*.

Chapter 2

Administering Content Server Applications

On the **Content Server Administration** page, the **Content Server Applications Administration** section allows you to manage Content Server applications.

2.1 Applications Management

The **Content Server Administration** > **Content Server Applications Administration** > **Applications Management** page allows you to build, rebuild, install, upgrade, uninstall, and delete applications. An application is a collection of Content Server nodes, support files, sub-tag files, and properties files. You can also use the **Applications Management** page to define a launch component or an initialization component for an existing application.

Installed applications will have a folder in the *<Content Server_home>* `\csapplications` directory on the server. Each application folder can contain the following items:

- **manifest file**

  Details the Content Server nodes and support files that make up the application.

- **XML dump file**

  Contains an export of the Content Server nodes included in the application. If the application does not contain any Content Server nodes, then it will not have an `XML dump` file.

- **applicationfiles folder**

  Contains the application's support files. If the application does not contain any application support files, then it will not have an `applicationfiles` folder.

  Note: In older applications, this folder was named `support`.

- **supportcollateral folder**

  Contains the support files from other support paths on Content Server. If the application does not contain such support files, then it will not have a `supportcollateral` folder.

- **subtags folder**

  Contains the application's sub-tag drop-in files. If the application does not contain sub-tag drop-in files, then it will not have a `subtags` folder.

- **properties folder**

  Contains the application's properties files. If the application does not contain properties files, then it will not have a `properties` folder.

The **Applications Management** page will show details of any installed or installable applications, along with options to perform various actions.

- To open the **Build Application** page, click the **Click to build a new application** link . For more information about how to build an application, see "Building or Rebuilding an Application" on page 50.
- To open the **Rebuild Application** page, on the row for a particular application, in the **Rebuild** column, click the **Rebuild** check box. For more information about how to rebuild an application, see "Building or Rebuilding an Application" on page 50.
- To see more details about an application, on the row for a particular application, in the **View Application** column, click the **View** link.

2.1.1 Building or Rebuilding an Application

The following table describes fields found in both the **Build Application** and the **Rebuild Application** pages:

Field Name	Mandatory?	Description
Application Name	Yes	You must give your application a name. Only alphanumeric characters (A-Z and 0-9) are allowed. All non-alphanumeric characters, such as hyphens, underscores, and spaces are not allowed.
Application Description	No	You can optionally give your application a description.
Version Number	Yes	You must give your application a version number. For example, "1.0".
Content Server Source Objects	No	You can optionally add Content Server nodes to your application. In the **Content Server Source Objects** box, use the **Browse For Source Object** button to browse to a Content Server source object. The Content Server source object will be exported, and will be imported into the target system during the installation process. If the Content Server source object is a container, all children of the object will also be included in the application.

Field Name	Mandatory?	Description
Application Files	No	You can optionally add application files to your application. Use the **Browse** button to browse for files.
Support Paths	No	You can optionally add support paths to your application. Enter a support path in the **Support Paths** box. For example: `foldername` or `foldername.txt`. The path should be under the `support` directory on Content Server, so if you enter "foldername", the `/img/foldername` folder will be included in the application. Paths can point to either folders or files. If a folder is selected then the folder and all of its contents will be added to the application.

Field Name	Mandatory?	Description
Sub-tags	No	You can optionally add sub-tag files to your application. In the **Sub-tags** box, use the **Browse** button to browse for files. Each sub-tag should consist of two files, a `subtagname.txt` file and a `subtagname.txt.json` file. If the **Tick to overwrite the existing sub-tag if present** check box is selected, then, when the application is installed, the sub-tag files attached to the application will be used. If they already exist on the system on which the application is being installed, the sub-tag files attached to the application will overwrite existing sub-tags. If the box is cleared and the sub-tag already exists on the target system, then the sub-tag files will not be installed. If any sub-tags attached to the application are disabled on the target system, they will be enabled during the installation process.
Module Dependencies	No	You can optionally add module dependencies to your application. In the **Module Dependencies** box, enter the name of a module. For example, if you type "WebReports" in the **Module Name** box, this will define the name of a module that must exist on the target system before the application can be installed. You can enter the **Version Number** and **Update Number** of the module dependency. If any module dependencies are missing on the target system, then an error will be thrown when attempting to install the application.

Field Name	Mandatory?	Description
Tag / Sub-tag Dependencies	No	You can optionally add tag or sub-tag dependencies to your application. In the **Tag / Sub-tag Dependencies** box, enter the name of a tag or sub-tag. For example, "LL_WEBREPORT_INSERTJSON" or "WFTASKINFO". This will define the name of a WebReports tag or sub-tag that must exist on the target system before the application can be installed. If the tag or sub-tag exists on the target system but is disabled, then it will be enabled during the installation process. If the tag or sub-tag is not found, then an error will be thrown when attempting to install the application.
ActiveView Override Type Dependencies	No	You can optionally add ActiveView override type dependencies to your application. In the **ActiveView Override Type Dependencies** field, select an ActiveView override type from the list. For example, click **Folder Browse**. This will define an ActiveView override type that must exist and be enabled on the target system before the application can be installed. If the ActiveView override type exists on the target system but is disabled, then it will be enabled during the installation process. If the ActiveView override type is not found, then an error will be thrown when attempting to install the application.

Field Name	Mandatory?	Description
INI Settings	No	You can optionally add INI settings to your application. These will be added to the `opentext.ini` file on the target system during the installation process. In the **INI Settings** area, in the **Section** list, click a section. For example, select "CSApps". Next, enter both a **Key** and a **Value**. This will define an INI Setting which will be written to the `opentext.ini` file. For example, if you select: • Section=CSApps • Key=MyKey • Value=x `MyKey=x` will be added to the `[CSApps]` section of the `opentext.ini` file. If the INI setting key already exists on the target system, then its value will be overwritten by the application's value during the installation process.
Properties Files	No	You can optionally add properties files to your application. An application can include properties files for different metadata languages. The naming convention of any properties files must be of the form *<application name>*_*<language code>*`.properties`. For example, "myapplication_en_US.properties". If any properties files are included, then one of them must be for `en_US`.

When you build or rebuild an application, the following actions will occur when you click **Submit**:

- When building a new application, a folder for the application will be created in the *<Content Server_home>*`\csapplications` directory on the server.

- When rebuilding an existing application, the folder for the application in the *<Content Server_home>*\csapplications directory on the server will be deleted and replaced with a new version reflecting the entries on the **Rebuild Application** page.
- If the application contains application files, then they will be copied to a folder in the support directory located at *<Content Server_home>*\support\csapplications*<application_name>*.
- If the application contains properties files, then they will be copied to a folder in the support directory located at *<Content Server_home>*\support\csapplications*<application_name>*\properties.
- When rebuilding an existing application, the folder for the application in the support directory on the server will be deleted, if it exists, and replaced with a new version reflecting the entries on the **Rebuild Application** page.

To Build or Rebuild an Application

To build or rebuild an application:

1. On the **Content Server Administration** page, in the **Content Server Applications Administration** section, click **Applications Management**.
2. On the **Applications Management** page, do one of the following:
   - To build a new application, select the **Click to build a new application** link.
   - To rebuild an existing application, select **Rebuild** next to that application's name.

   Note: The **Rebuild Application** page for the application will be pre-populated with data from the existing build of the application.
3. On the **Build Application** page or the **Rebuild Application** page, in the **Application Name** field, enter, or change, the name of this application. Application names must consist only of alphanumeric characters.
4. Optional In the **Application Description** field, enter or change the description of this application.
5. In the **Version Number** field, enter, or change, the version number for this application. For example, "1.0".
6. Optional In the **Content Server Source Objects** box, click **Browse For Source Object** and select one or more Content Server nodes to add to your application.
7. Optional In the **Support Files** box, click **Browse** and select one or more files to add to your application.

   Note: If you are rebuilding an existing application, any application files that are already present in the application will be listed in the **Existing Application Files** area.

8. Optional In the **Support Paths** box, enter one or more support paths to add to your application.

   Note: If you are rebuilding an existing application, any supports paths that are already present in the application will be listed in the **Existing Support Paths** area.

9. Optional To add one or more sub-tag files to your application, in the **Sub-tags** area, do the following:

   a. In the first, upper text field, enter a *<subtagname>*.txt file name.

   b. In the second, lower text field, enter a *<subtagname>*.txt.json file name.

   c. Optional If you want to ensure that the sub-tag files attached to the application overwrite any existing sub-tags, select the **Tick to overwrite the existing sub-tag if present** check box.

   Note: If you are rebuilding an existing application, any sub-tags that are already present in the application will be listed in the **Existing Sub-tags** area.

10. Optional To define the name of a module that must exist on the target system before the application can be installed, in the **Module Dependencies** area, add one or more module dependencies to your application:

    a. In the **Module Name** field, enter the name of a module.

    b. In the **Version Number** field, enter the version number of the module.

    c. In the **Update Number** field, enter the update number of the module.

11. Optional In the **Tag / Sub-tag Dependencies** box, enter the name of one or more tags or sub-tags to add the dependencies to your application. For example, type `LL_WEBREPORT_INSERTJSON` or `WFTASKINFO`.

    Note: If the tags or sub-tags are not found on the target system, the application will not be installed.

12. Optional In the **ActiveView Override Type Dependencies** list, click to add one or more ActiveView override type dependencies to your application.

    Note: If the ActiveView override type(s) is not found on the target system, the application will not be installed.

13. Optional In the **INI Settings** area, add one or more INI settings to your application:

    a. In the **Section** list, click an INI section selection.

    b. In the **Key** field, enter a key name.

    c. In the **Value** field, enter a value for the key.

Note: If the INI setting(s) already exist(s) on the target system, its/their value will be overwritten by the application's value during the installation process:

14. Optional In the **Properties Files** box, click **Browse** and select one or more properties files to add to your application.

Notes

- If you include any properties files, at least one of them must be for `en_US`.
- If you are rebuilding an existing application, any properties files that are already present in the application will be listed in the **Existing Properties Files** area.

15. Click **Submit**.

2.1.2 Installing an Application

When you install an application, the following actions will occur after you click **Submit**:

1. If the application contains Content Server nodes, the application's `XML dump` file are imported, creating the Content Server nodes for that application in the selected target container.
2. If the application contains application files, those files are copied to a folder in the support directory at *<Content Server_home>*`\support\csapplications\`*<application_name>*.
3. If the application contains support paths, then the paths are copied to equivalent locations in the support directory. Only files that do not already exist in the support directory will be copied. Any files that are already present in the support directory will not be overwritten with the files included in the application.
4. If the application contains properties files, those files are copied to a folder in the support directory at *<Content Server_home>*`\support\csapplications\`*<application_name>*`\properties`.
5. Other actions are performed depending on the application definition. Use the **Preview** option for more details.
6. The application's manifest file is replaced with a new version. If the application contains Content Server nodes then the manifest file will contain the node IDs that have been created on the target system.
7. The application folder is moved from *<Content Server_home>*`\csapplicationsstaging` to *<Content Server_home>*`\csapplications`.

To Install an Application

To install an application:

1. Copy the application folder to the *<Content Server_home>* `\csapplicationsstaging` directory on the server.
2. On the **Content Server Administration** page, in the **Content Server Applications Administration** section, click **Applications Management**.
3. On the **Applications Management** page, on the row for the application that you want to install, select the **Install** check box.
4. On the **Install Application** page, do one or more of the following:
   a. Optional If the application contains Content Server nodes, then the form contains the **Target Container** field. To browse to a Content Server source object where the Content Server nodes for the application will be installed, click **Browse For Container** .
   b. Optional To see details of actions that will be performed on the target system by the installation process, click **Preview** .
   c. Click **Submit**.

2.1.3 Upgrading an Application

When you upgrade an application, the following actions occur when you click **Submit**:

1. The application folder for the currently-installed version of the application is deleted from *<Content Server_home>*`\csapplications`.
2. The Content Server nodes for the existing application is deleted, except for any nodes that have been specified not to be deleted.
3. If the application contains Content Server nodes then the application's `XML dump` file is imported, creating the application's Content Server nodes in the selected target container. This will default to the same location where the application was originally installed, unless a different location is selected.

   Note: This process will fail if the import process attempts to create any nodes with the same name as nodes that already exist in the target container.
4. If the application contains application files, they are copied to a folder in the support directory at *<Content Server_home>*`\support\csapplications\`*<application_name>*.
5. If the application contains support paths, then the paths are copied to equivalent locations in the support directory. Only files that do not already exist in the support directory are copied. Any files that are already present are not overwritten with the files included in the application.

6. If the application contains properties files then they are copied to a folder in the support directory at *<Content Server_home>*\support\csapplications*<application_name>*\properties.
7. Other actions are performed depending on the application definition. Use the **Preview** option for more details.
8. The application's manifest file is replaced with a new version. If the application contains Content Server nodes then the manifest file will contain the node IDs that have been created on the target system.
9. The application folder for the new version of the application is moved from *<Content Server_home>*\csapplicationsstaging to *<Content Server_home>*\csapplications.

To Upgrade an Application

To upgrade an application:

1. Copy a higher version of the application folder than is already installed to the *<Content Server_home>*\csapplicationsstaging directory on the server.
2. On the **Content Server Administration** page, in the **Content Server Applications Administration** section, click **Applications Management**.
3. On the **Applications Management** page, select the **Upgrade** check box beside the name of the application you want to upgrade.
4. On the **Upgrade Application** page, do one or more the following:

   a. Optional If the application contains Content Server nodes that have been renamed, been moved, or had a new version added to them since the application was installed, the nodes will be listed in the **Nodes that have been altered since the application was installed** table.

      To delete the listed nodes when the application is upgraded, select the **Delete** box next to the nodes you want deleted.

   b. Optional If the application contains Content Server nodes that have remained unchanged since the application was installed, the nodes will be listed in the **Nodes that have been unaltered since the application was installed** table.

      To delete a node from this table, when the application is upgraded, select the **Delete** check box next to the nodes you want deleted.

      Note: Any Content Server nodes that have been deleted since the application was installed will be listed in the **Nodes that have been deleted since the application was installed** table.

   c. Optional To select a target location for the new version of the application, select the **Tick to install a new version of the application in a different location** check box.

      If this box is cleared then the application will install to the same location as the previous version.

d. Optional To see details of actions that will be performed on the target system by the installation part of the upgrade process, click **Preview** .

e. Click **Submit**.

2.1.4 Uninstalling or Deleting an Application

When you *uninstall* an application, the following actions will occur after you click **Submit**:

1. The application folder is moved from *<Content Server_home>*\csapplications to *<Content Server_home>*\csapplicationsstaging.
2. If a support folder exists for the application, it is deleted.
3. The Content Server nodes for the application are deleted, except for any nodes that have been specified not to be deleted.

When you *delete* an application, the following actions will occur when you click **Submit**:

1. The application folder is deleted from *<Content Server_home>*\csapplications.
2. If a support folder exists for the application, it is deleted.
3. The Content Server nodes for the application are deleted, except for any nodes that have been specified not to be deleted.

To Uninstall an Application

To uninstall an application:

1. On the **Content Server Administration** page, in the **Content Server Applications Administration** section, click **Applications Management**.
2. On the **Applications Management** page, on the row for the application that you want to uninstall, select the **Uninstall** check box. This will first uninstall the application and will then move the application folder for the uninstalled application into the *<Content Server_home>*\csapplicationsstaging directory.
3. Optional On the **Uninstall Application** page, do one or more of the following additional actions:

   - If the application contains Content Server nodes that have been renamed, been moved, or had a new version added to them since the application was installed, the nodes will be listed in the **Nodes that have been altered since the application was installed** table.

     To delete a node from this table when the application uninstalls, select the **Delete** check box next to the nodes that you want deleted.

   - If the application contains Content Server nodes that have are unchanged since the application was installed, the nodes will be listed in the the **Nodes that have been unaltered since the application was installed** table.

To delete a node from this table when the application is uninstalled, select the **Delete** check box next to the nodes that you want to delete.

Note: Any Content Server nodes that have been deleted since the application was installed will be listed under **Nodes that have been deleted since the application was installed**.

4. Click **Submit**.

To Delete an Application

To delete an application:

1. On the **Content Server Administration** page, in the **Content Server Applications Administration** section, click **Applications Management**.
2. On the **Applications Management** page, on the row for the application that you want to delete, select the **Delete** check box. This will first uninstall the application and will then delete the application folder for the uninstalled application from the *<Content Server_home>*\csapplicationsstaging directory.
3. Optional On the **Delete Application** page, do one or more of the following additional actions:
   - If the application contains Content Server nodes that have been renamed, been moved, or had a new version added to them since the application was installed, the nodes will be listed in the **Nodes that have been altered since the application was installed** table.

     To delete a node from this table when the application uninstalls, select the **Delete** check box next to the nodes you want deleted.
   - If the application contains Content Server nodes that are unchanged since the application was installed, the nodes will be listed in the **Nodes that have been unaltered since the application was installed** table.

     To delete a node from this table when the application is deleted, select the **Delete** check box next to the nodes you want deleted.

     Note: Any Content Server nodes that have been deleted since the application was installed will be listed under **Nodes that have been deleted since the application was installed**.
4. Click **Submit**.

2.1.5 Define Special Components

The **Define Special Components** form will list all the WebReports that are part of the application. The form can be used to:

- Define a default launch component for the application.
- Define an initialize component for the application.
- Assign unique nicknames to WebReports that are part of the application.

Default Launch Component

An application can be launched using the URL: `?func=csapps.launchapp&appname=<application_name>`. Calling this URL will run the WebReport defined as the default launch component.

Initialize Component

A WebReport defined as the initialize component runs immediately after the application is installed.

Unique Nicknames

WebReports can be given unique nicknames within an application and then run using the URL: `?func=csapps.launchapp&appname=<application_name>&nickname=<unique_nickname>`

Form Usage

The form will list all the WebReports that are part of the application:

1. Do one of the following:

   - If you want to define a WebReport as the default launch component for the application, select the radio button beside that WebReport.
   - If you don't want the default option, which is to define a launch component for the application, select the radio button next to **No Component selected for this function**.

2. Enter unique nicknames in the field to the next to each WebReport.

   The nicknames will be validated on submission of the form. An error will be generated if a nickname is not unique within the application.

   Note: Nickname comparisons are case-insensitive. For example, “mynickname” is the same as “MyNickName” and will cause a validation error.

3. Click **Submit**.

2.2 Applications Management Configuration

Use the **Content Server Administration** > **Applications Management Configuration** page to manage which directories are used by "Applications Management" on page 49.

Use the **Content Server Applications Directory** box to set which directory on the server is used to store folders for installed applications.

Use the **Content Server Applications Staging Directory** box to set which directory on the server is used to store folders for applications to be installed or upgraded, or that have been uninstalled. The directories specified in the **Content Server Applications Directory** and **Content Server Applications Staging Directory** field should already exist on the server.

Clicking **Submit** validates the fields to ensure that the values entered are legitimate directories. Clicking **Reset** clears any changes made since the page was last refreshed.

2.2.1 To Set Directories Used by Content Server Applications Management

To set directories used by Content Server applications management:

1. On the **Content Server Administration** page, in the **Content Server Applications Administration** section, click **Applications Management Configuration**.
2. On the **Applications Management Configuration** page, in the **Content Server Applications Directory** box, type the location of the Content Server applications. The default is *<Content Server_home>*`\csapplications\`.
3. In the **Content Server Applications Staging Directory** box, enter the staging location for the Content Server applications. The default is *<Content Server_home>*`\csapplicationsstaging\`
4. In the **Content Server Applications Support Folder Name** box, enter the folder name for the Content Server applications. The default is `csapplications`.
5. Click **Submit**.
6. Restart Content Server.

2.3 Applications Volume

Use the **Content Server Administration** > **Content Server Applications** page to work with the objects in the Applications Volume.

The **Content Server Applications Volume** is the default destination for applications built or installed using "Applications Management" on page 49.

Chapter 3

Content Intelligence Widget Configuration

This section describes the use and configuration for the Content Intelligence widgets.

Note: You will need to have a valid WebReports license to see the Content Intelligence widgets.

- Nodes List WebReport widget. For more information, see "Configuring the Nodes List WebReport Widget" on page 65.
- HTML WebReport widget. For more information, see "Configuring the HTML WebReport Widget" on page 67.
- Widget Carousel widget. For more information, see "Configuring the Widget Carousel Widget" on page 82.
- Visual Count widget. For more information, see "Configuring the Visual Count Widget" on page 85.

To use the Content Intelligence widgets, you must include and configure each widget in a perspective. For more information about how to create a perspective, see the *Online Help* available in Perspective Manager.

Note: Any WebReport that you intend to display in the Smart UI cannot contain custom parameters. Custom parameters can only be displayed in the Classic UI. By default, any custom parameter screen executed from the Smart UI will run in the Classic UI. A custom parameter is created on the **Parameters** tab when you select "Custom" from the **Type** list.

3.1 Configuring the Nodes List WebReport Widget

The **Nodes List WebReport** widget shows an expandable list of data, such as a list of documents that share a common metadata tab, based on a specified WebReports data source. Similar to the standard Content Server Favorites widget, the **Nodes List WebReport** widget supports scrolling and name filtering. This widget uses the WebReports REST API along with the `INSERTJSON` content control tag and the `@NODESTABLEFIELDS` directive to call the specified WebReport based on the `widgets_nodeslist_nodestable` default reportview to return correctly formatted JSON data. Additional tags within the template support sorting, filtering, formatting, and pagination, and respond to parameters passed in by the client. For more information about the `@NODESTABLEFIELDS` directive, and other directives that specify WebReports Smart UI reports output, see the INSERTJSON tag.

End users can expand the tile containing the **Nodes List WebReport** widget to see a full table view of the nodes. When expanded, the nodes table shows additional

columns of data, such as Type, Name, Size, and Modified. Similar to the standard Smart View browse view, the expanded widget allows users to filter on name, sort by column, and view the properties for each node.

Note: The expanded **Nodes List WebReport** widget does not replicate all Smart View browse behavior. For example, the columns shown are only a static subset of the default: neither the facet bar, nor the multi-action bar is shown.

After you have included the **Nodes List WebReport** widget in your perspective, you can configure the following parameters:

Nodes List WebReport Widget Configuration Parameters

Name	Description
Title	Mandatory. Enter the title for the tile. Typically, this would describe the WebReport that you are rendering.
Icon Class	Optional. Provide the CSS class for the icon that you want to appear in the top left corner. For example: *<Content ServerInstallDir>*/`support/csui/themes/carbonfiber/icons.css` contains icons such as `title-assignments`, `title-customers`, `title-favourites`, `title-opportunities`, `title-recentlyaccessed`, `title-activityfeed`, `title-customviewsearch`. Default = title-webreports 
Search Placeholder	Optional. Enter a custom string that will appear when the user clicks **Search**. Default value = "Search NodesList Report."
WebReport ID	Mandatory. Enter the ID for the WebReport that you want to appear on the tile. **Note:** The WebReport provided must be based on the default WebReports template for `widget_nodeslist_nodestable`.
WebReports Parameters	Optional. Enter one or more "`name`"-"`value`" pairs for the parameters that you want to pass into the WebReport.

3.1.1 To Configure the Nodes List WebReport Widget

To configure the Nodes List WebReport widget:

1. Create a new WebReports node using the `widget_nodeslist_nodestable` default reportview. For information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.
2. On the **Source** tab, set the data source of the WebReport to the database source that contains a column called DataID, which references valid nodes that you want to display in the widget. For more information about how to set the data source, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.
3. On the **Content Server Administration** > **ActiveView Administration** page, click the **Open the Perspective Manager** link.

4. In the Perspective Manager, create a new perspective or edit an existing perspective. For more information about how to create a new perspective or how to edit a perspective, see the *Online Help* available in the Perspective Manager.

5. On the **Configure** tab, do the following:

   a. In the **Widget Library** pane, click to expand the **Content Intelligence** widget group and drag the **Nodes List WebReport** widget to the page.

   b. To open the configuration options, select the **Nodes List WebReport** widget on the page.

   c. In the **Options** pane, in the **WebReport ID** box, add the WebReport created in Step 1 and edited in Step 2.

   Note: For more information on the options available for the **Nodes List WebReport** widget, see "Nodes List WebReport Widget Configuration Parameters" on page 66.

6. Navigate to the location where the new perspective is applied to verify the **Nodes List WebReport** widget output.

3.2 Configuring the HTML WebReport Widget

The HTML WebReport widget allows developers to show formatted text and images in a Smart View widget. This widget calls a specified WebReport using the WebReports Rest API and inserts the HTML output of the WebReport into the tile at runtime.

Notes

- This widget supports HTML and CSS. For the 16.0.3 release and onwards, this widget will also support Javascript. For releases prior to 16.0.3, Javascript is *not* supported.
- Be sure to avoid style conflicts with the Content Server Smart View user interface framework or any other widgets on the page.

The following sample reportview templates provide examples of how to construct the HTML output of the WebReport content to avoid conflicts. They show how to use CSS selectors to avoid style conflicts, images and icons, HTML tables and dynamic data insertion using WebReports tags, including the new `[LL_REPTAG_WIDGETCONTAINERID /]` data tag which can be used to obtain the ID current container. They also include examples of how to make page elements responsive. It is important to avoid including unresponsive content. For instance, inserting an image with a fixed width will mean that when the page is resized or viewed on a mobile device the page flow may not adjust as desired.

- `widget_html_report_image_icons_sample`
- `widget_html_report_responsive_table_sample`

Important

Although these templates include examples using the bootstrap library, OpenText does not officially support this library. If you use this library, you must consult the published documentation. In addition, the Content Server Smart View implements a customized version of bootstrap. It is expected that in some cases the Smart View version of bootstrap will diverge from the publicly documented behavior. In a post-Content Server 16 update, OpenText will provide developer documentation with style guidelines as well as the Smart View UI SDK.

After you have included the HTML WebReport widget in your perspective, you can configure the following parameters:

HTML WebReport Widget Configuration Parameters

Name	Description
Widget Size	Mandatory. Select the size for the HTML WebReport widget in the Flow layout: Single width, Double width, or Full Page. Default = Double width.
Title	Optional. Enter the title for the tile. Typically, this would describe the WebReport that you are rendering. Default title = HTML WebReport
Icon Class	Optional. Provide the CSS class for the icon that you want to appear in the top left corner. For example: `support/csui/themes/carbonfiber/icons.css` contains icons such as `title-assignments`, `title-customers`, `title-favourites`, `title-opportunities`, `title-recentlyaccessed`, `title-activityfeed`, `title-customviewsearch`. Default = title-webreports  **Note:** You can specify a custom icon within the WebReport template by adding a CSS selector that uses `background-image` and then referencing the selector in this option.
Header	Optional. Select the option for whether the tile will include a header area that contains the title and title icon. Default = True.
Scroll	Optional. Select the option for whether users can scroll through the content on the tile. If scrolling is disabled, content will be truncated. Default = True.

WebReport ID	Mandatory. Enter the ID for the WebReport that you want to appear on the tile. **Note:** The WebReport provided must follow the HTML construction guidelines and be based on either a blank template or one of the sample WebReports templates, `widget_html_report_image_icons_sample` or `widget_html_report_responsive_table_sample`.
WebReports Parameters	Enter one or more "`name`"-"`value`" pairs for the parameters that you want to pass into the WebReport.

3.2.1 To Configure the HTML WebReport Widget

To configure the HTML WebReport widget:

1. Create a new WebReport node using the blank reportview, the `widget_html_report_image_icons_sample` template, or the `widget_html_report_responsive_table_sample` template. For information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

2. Edit the WebReport to contain the desired output using the guidelines and general practices used in the sample reportviews.

3. On the **Content Server Administration** > **ActiveView Administration** page, click the **Open the Perspective Manager** link.

4. In the Perspective Manager, create a new perspective or edit an existing perspective. For more information about how to create a new perspective or how to edit a perspective, see the *Online Help* available in the Perspective Manager.

5. On the **Configure** tab, do the following:

   a. In the **Widget Library** pane, click to expand the **Content Intelligence** widget group and drag the HTML WebReport widget to the page.

   b. To open the configuration options, select the **HTML WebReport** widget on the page.

   c. In the **Options** pane, in the **WebReport ID** field, add the WebReport created in Step 1 and edited in Step 2.

      Note: For more information on the options available for the **HTML WebReport**widget, see "HTML WebReport Widget Configuration Parameters" on page 68.

6. Browse to the location where the new perspective is applied to verify the HTML WebReport widget output.

3.2.2 Using Javascript in the HTML WebReport widget

The HTML WebReport widget supports Javascript. It is important, however, to understand which types of syntax are supported and which are not. The Smart View is a single page application that uses RequireJS modules to load Javascript. All libraries and components that are available for use within the official Smart UI SDK are also available within the HTML WebReport widget. To use these libraries and components, you must use the correct RequireJS syntax to load modules.

Important

Javascript used with the HMTL WebReport widget must follow the standards described in this help and in the Smart UI SDK documentation. Javascript that is not written to be compatible with the Smart UI may conflict with the Smart UI framework and result in unpredictable behavior.

Watch out for Javascript in your HTML WebReport widget that is not written to be compatible with the Smart UI. The following situations may cause problems:

- If you include a WebReport that is written for the Classic UI, it may include Javascript that conflicts with the Smart UI framework.
- If you use `<script scr="..." >` syntax to load external Javascript files directly on the page, such as new third-party libraries, these libraries may conflict with the Smart UI framework. Including such an HTML WebReport widget on a Smart UI page that is subsequently embedded in an external site may also cause problems.

Loading Smart UI SDK Modules

Note: The Smart UI SDK is available starting from the 16.0.3 Content Server release, so this approach is not supported for releases prior to 16.0.3.

If you add a script block to the WebReport that will be loading within an HTML WebReport widget, you can access the `csui` global object. Then, you can use `csui.require` to load modules and use of them in your application.

Example 3-1: RequireJS Example Syntax

```
<script>

    // Wrap any calls to third party modules and Smart UI
components using csui.require.
                csui.require(['csui/lib/jquery', 'csui/utils/
base'], function ($, base) {

                                  // The 'jQuery' third party
libraries is now loaded and can be accessed by the '$'
reference. e.g.
                                  $('.myclass').show();

                                  //The Smart UI 'base' utility
is also loaded and can be accessed by the 'base' reference.
```

```
    e.g.
                                        base.isTouchBrowser();

        });

    </script>
```

Commonly Used Third-Party Libraries

- **Backbone**: csui/lib/backbone
- **Handlebars:** csui/lib/handlebars
- **jQuery**: csui/lib/jquery
- **Marionette**: csui/lib/marionette
- **Underscore**: csui/lib/underscore

3.2.3 Example Scripts for the HTML WebReport widget

This section includes example scripts for the HTML WebReport widget.

"Example 1: Using Standard Javascript" on page 71

"Example 2: Using JQuery" on page 72

"Example 3: Using Marionette" on page 72

Example 1: Using Standard Javascript

The following script shows the use of standard Javascript:

```
<style>

    .binf-widgets .webreports-example-hello {
        margin: 10px;
        padding: 10px;
    }

</style>

<div class="webreports-example-hello"></div>

<script>

    document.getElementsByClassName("webreports-example-hello")
[0].style.border = "2px solid #87CEFA";
    document.getElementsByClassName("webreports-example-hello")
[0].style.borderRadius = "10px";
    document.getElementsByClassName("webreports-example-hello")
[0].innerHTML = "Hello World";
```

```
</script>
```

Example 2: Using JQuery

The following script shows the use of JQuery:

```
<style>

    .binf-widgets .webreports-example-hello-jquery {
        margin: 10px;
        padding: 10px;
    }

</style>

<div class="webreports-example-hello-jquery"><div>

<script>

    csui.require(['csui/lib/jquery'], function ($) {

    $(".webreports-example-hello-jquery").css("border","2px solid
#87CEFA");
    $(".webreports-example-hello-jquery").css("border-
radius","10px");
    $(".webreports-example-hello-jquery").html("Hello World");

    });

</script>
```

Example 3: Using Marionette

The following example shows how to use Marionette, the main library used for rendering HTML in the Smart View:

```
<style>

    .binf-widgets .webreports-example-marionette-view {
        margin: 10px;
        padding: 10px;
        border: 2px solid #87CEFA;
        border-radius: 10px;
    }

</style>

<script>

    csui.require(['csui/lib/jquery', 'csui/lib/marionette'],
function ($, Marionette) {

        var helloWorldView,
```

```
            // Create a Marionette Region from our main div element
            contentRegion = new Marionette.Region({el: '#webreports-
example-marionette'}),
            // Create a new Marionette view extending from the built-
in ItemView
            HelloWorldView = Marionette.ItemView.extend({
              // Apply a class name to the view element
              className: 'webreports-example-marionette-view',
              // Override the render function to update the view.
              render: function () {
                this.$el.html('Hello World');
                return this;
              }
            });

        // Create a new instance of our view
        helloWorldView = new HelloWorldView();
        // Show our view in the region
        contentRegion.show(helloWorldView);

    });

 </script>

 <div id="webreports-example-marionette"></div>
```

3.2.4 Instructions to create a WebReport to return data for a Smart View TABLEREPORT

This section includes step-by-step instructions to create an HTML WebReport widget for Smart View that returns data using the `@TABLEREPORT` directive. The `@TABLEREPORT` directive enables you to specify how data is sorted, paged, and filtered.

Collection processing refers to the paging, sorting, and filtering for the data. It can be one “webreport” or “datasource”. If “webreport” is specified then all the data from the data source will be returned to Content Server and collection processing will be performed by the WebReports engine. If “datasource” is specified then collection processing will be performed by the WebReports data source.

For more information, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

Method 1: Instructions to perform collection processing in the WebReport

This example walks through how to create an HTML WebReport widget that performs collection processing in WebReport using the `@TABLEREPORT` directive. For more information about the `@TABLEREPORT` directive, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

Important

Because this method does not perform pagination in the data source, all rows will be returned to the WebReport for processing. Therefore, OpenText recommends that you only apply this method for small to medium sized data sets.

If you have a large data set, see "Method 2: Instructions to perform collection processing in the data source" on page 77

To create an HTML WebReport widget that performs collection processing in the WebReport:

1. Create a WebReport:
   a. In Content Server, click **Add Item**, and then click WebReport.
   b. On the **Add WebReport** page, in the **Name** section, enter a unique name for the WebReport. For the purposes of this example, call your new WebReport "WebRep CPWR01".
   c. From the **Reportview File** list, from the list, click **widget_table_report_process_data_in_webreport**.
   d. In the **Data Source** section click **Browse Content Server**.

      Find, or create, a Content Server folder that contains multiple documents. Click the **Select** link next to that folder. You will use this Content Server folder node as a data source.

      This example will refer to the folder as "CPWR Folder01".
   e. Click **Add**.
   f. From your new WebReport's **Functions** menu, select **Properties** > **General**. On the **General** tab, copy the value in the **Nickname** field.
2. Edit the WebReport you created in Step 1, "WebRep CPWR01".
   a. From the **Functions** menu of the "WebRep CPWR01" WebReport, click **Edit Reportview**.
   b. In the **Edit WebReport** page, scroll to the bottom. Edit the bottom section so that it reads:

```
[LL_WEBREPORT_INSERTJSON @TABLEREPORT
COLLECTIONPROCESSING:"webreport"
INCLUDECOLUMNS:'["Name","MimeType", "SubType", "ModifyDate",
"UserID"]'
```

```
FORMATCOLUMNNAMES:MimeType:"Mime Type":SubType:"Sub
Type":UserID:"Owner"
FORMATCOLUMNS:SubType:"[LL_REPTAG=SubType
LABEL:SUBTYPE /]":CreatedBy:"[LL_REPTAG=CreatedBy
USERINFO:FULLNAME /]":ModifyDate:"[LL_REPTAG=ModifyDate
DATE:LONG /]" /]
```

c. Click **Add Version**.

3. Create a second new WebReport. For the purposes of this example, call your second new WebReport "WebRep CPWR02":

   a. In Content Server, from the **Add Item** menu, click **WebReport**.

   b. In the **Name** field type "WebRep DPWR02".

   c. From the **Reportview File** list, click **blank_report**.

   d. Click **Add**. In the information message dialog informing you that you have not selected a data source, click **OK**.

   e. From the **Functions** menu of the "WebRep CPWR02" WebReport, click **Edit Reportview**.

   f. In the **Edit WebReport** page, edit the content so that it reads:

```
[/* Title: Blank Reportview */]

<script>
    csui.require(['csui/lib/marionette', 'csui/utils/
contexts/page/page.context', 'webreports/controls/
table.report/table.report.view'], function (Marionette,
PageContext, TableReportView) {

         var contentRegion = new Marionette.Region({el:
'#content'}),
             pageContext   = new PageContext(),
             tableReportView,
             options;

         options = {
            context: pageContext,
            data: {
                id: 463381,
                title: 'My Table Report',
                header: false,
                titleBarIcon: 'title-assignments',
                columnsWithSearch: 'name',
                sortBy: 'ModifyDate',
                sortOrder: 'desc'
            }
         };

         tableReportView = new TableReportView(options);
         contentRegion.show(tableReportView);
         pageContext.fetch();

    });
```

```
</script>

<div id="content"></div>

[LL_WEBREPORT_STARTROW /]
[LL_WEBREPORT_ENDROW /]
```

g. Before you save your changes, edit the line:

```
id: 463381,
```

to put in the ID that you copied from the first WebReport in Step 1.f.

h. Click **Add Version**.

4. Open the **Perspective Manager** in the Smart UI and create a new perspective. Choose, or create, a specific test folder in which to save this perspective. For the purposes of this example, browse to "WebRep TableReport Perspective".

   For information about the **Perspective Manager**, see the *Online Help* available in Perspective Manager.

5. Configure your new perspective:

   a. On the **General** tab:

      - Type a unique name for this perspective. For the purposes of this example, call this perspective "WR-Persp01".

   b. On the **Layout** tab, in the **Type** field, click "Left - Center - Right".

   c. On the **Configure** tab:

      i. Under the **Content Intelligence** widget group, select the "HTML WebReport" widget and drag it on the perspective, in the Center position.

      ii. You need to choose a WebReport for the **WebReport ID** field. In the right hand side panel, in the "HTML WebReport" widget options, click **Browse**. Choose the WebReport you created in Step 3. For this purposes of this example, choose "WebRep CPWR02".

   d. Create the perspective.

6. While in Smart View, browse to the test folder in which you saved the perspective in Step 4. For the purposes of this example, browse to "WebRep TableReport Perspective".

Method 2: Instructions to perform collection processing in the data source

This example walks through how to create an HTML WebReport widget that shows a Table Report using collection processing in the data source with the `@TABLEREPORT` directive. For more information about the `@TABLEREPORT` directive, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*. This approach can be used to support larger data sets. The data source for this needs to be configured in a specific way to allow paging, sorting and filtering to be done correctly. Currently, only LiveReport data sources are supported for this approach.

Important

Because this method performs pagination in the data source, only one page of results will be returned to the WebReport for processing. Therefore, this method can be used for larger data sets.

If you have a small to medium sized data set, see "Method 1: Instructions to perform collection processing in the WebReport" on page 74

To create an HTML WebReport widget that performs collection processing in data source:

1. Create a WebReport:
   a. In Content Server, click **Add Item**, and then click WebReport.
   b. On the **Add WebReport** page, in the **Name** section, enter a unique name for the WebReport.

      For the purposes of this example, call your new WebReport "WebRep CPDS01".
   c. From the **Reportview File** list, click **widget_table_report_process_data_in_datasource**.
   d. From the **LiveReport File** list click "table_report_audit_events".

      In addition to creating a WebReport, this will also create a new LiveReport node. This LiveReport node will be called "Audit Events for This Week". It will also automatically set the LiveReport as the data source.
   e. Click **Add**.
   f. From your new WebReport's **Functions** menu, select **Properties** > **General**. On the **General** tab, take note of the value in the **Nickname** field. You will need this value later in this procedure in Step 4.g.
2. Choose, or create, a sample Content Server folder node to use as a container. For the purposes of this example, create a folder called "DatSrc TableReport Perspective".
3. Edit the WebReport you created in Step 1, "WebRep CPDS01".
   a. From the **Functions** menu of the "WebRep CPDS01" WebReport, click **Edit Reportview**.

b. In the **Edit WebReport** page, scroll to the bottom. Edit the bottom section so that it reads:

```
[LL_WEBREPORT_INSERTJSON @TABLEREPORT
COLLECTIONPROCESSING:"datasource"
EXCLUDECOLUMNS:'["RowNumber"]'
FORMATCOLUMNNAMES:AuditID:"Event":PerformerID:"User"
FORMATCOLUMNS:AuditID:"[LL_REPTAG=AuditID
LABEL:AUDITID /]":SubType:"[LL_REPTAG=SubType LABEL:SUBTYPE
DEF:'' /]":PerformerID:"[LL_REPTAG=PerformerID
USERINFO:NAME /]" /]
```

Note: If `FORMATCOLUMNS` is used with `COLLECTIONPROCESSING` set to "datasource", sorting will be automatically disabled for the formatted columns. This is because the sort values in the user interface are not available for sorting in the LiveReport. If you need formatted values that are also sortable you should do the formatting in the SQL.

c. Click **Add Version**.

4. Create a second new WebReport. For the purposes of this example, call your second new WebReport "WebRep CPDS02":

   a. In Content Server, from the **Add Item** menu, click **WebReport**.

   b. In the **Name** field type "WebRep DPDS02".

   c. From the **Reportview File** list, click **blank_report**.

   d. Click **Add**. In the information message dialog informing you that you have not selected a data source, click **OK**.

   e. From the **Functions** menu of the "WebRep CPDS02" WebReport, click **Edit Reportview**.

   f. In the **Edit WebReport** page, edit the content so that it reads:

```
[/* Title: Blank Reportview */]

<script>
    csui.require(['csui/lib/marionette', 'csui/utils/
contexts/page/page.context', 'webreports/controls/
table.report/table.report.view'], function (Marionette,
PageContext, TableReportView) {

         var contentRegion = new Marionette.Region({el:
'#content'}),
             pageContext   = new PageContext(),
             tableReportView,
             options;

         options = {
            context: pageContext,
            data: {
                id: 463411,
                title: 'Audit Events for the Past Week',
                header: false,
```

```
                titleBarIcon: 'title-assignments',
                columnsWithSearch: 'name',
                sortBy: 'AuditDate',
                sortOrder: 'desc'
            }
        };

        tableReportView = new TableReportView(options);
        contentRegion.show(tableReportView);
        pageContext.fetch();

    });
</script>

<div id="content"></div>

[LL_WEBREPORT_STARTROW /]
[LL_WEBREPORT_ENDROW /]
```

g. Before you save your changes, edit the line:

```
id: 463411,
```

to put in the ID that you noted from the first WebReport in Step 1.f.

h. Click **Add Version**.

5. Open the **Perspective Manager** in the Smart UI and create a new perspective.

   For information about the **Perspective Manager**, see the *Online Help* available in Perspective Manager.

6. Configure your new perspective:

   a. On the **General** tab:

      i. Type a unique name for this perspective. For the purposes of this example, call this perspective “DS-Persp01”.

      ii. Choose a specific test folder in which to save this perspective. You can choose to save this perspective in the folder you selected in Step 2. For the purposes of this example, browse to “DatSrc TableReport Perspective”.

   b. On the **Layout** tab, in the **Type** field, click “Left - Center - Right”.

   c. On the **Configure** tab:

      i. Under the **Content Intelligence** widget group, select the “HTML WebReport” widget and drag it on the perspective, in the Center position.

      ii. You need to choose a WebReport for the **WebReport ID** field. In the right hand side panel, in the “HTML WebReport” widget options, click **Browse**. Choose the WebReport you created in Step 4. For this purposes of this example, choose “WebRep CPDS02”.

   d. Create the perspective.

7. While in Smart View, browse to the test folder in which you saved the perspective. For the purposes of this example, browse to "DatSrc TableReport Perspective".

8. Optional If you want, you can choose to modify the report to return different data.

   Important

   Following this next step requires a strong knowledge of SQL and LiveReports.

   This data source is just a sample to get you started. It is set up in a specific way to work with the `INSERTJSON @TABLEREPORT` option when `COLLECTIONPROCESSING` is set to "datasource". For more information, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

   To modify the report to return different data, do the following:

   a. Rename the LiveReport that was automatically created in Step 1 to "Documents Modified in the Past Week". Edit the LiveReport "Documents Modified in the Past Week". In the **Title** field, type "Documents Modified in the Past Week".

   b. In the **SQL** section, edit the "ResultSetSubQuery" SQL statement.

      Remove this section of code:

```
        SELECT DAuditNew.AuditID AS AuditID,
DAuditNew.AuditDate AS AuditDate, DTree.Name AS Name,
DAuditNew.SubType AS SubType, DAuditNew.PerformerID AS
PerformerID
        FROM DAuditNew
        INNER JOIN DTree ON DAuditNew.DataID = DTree.DataID
        WHERE DAuditNew.AuditDate > %7
        AND %8
        ~3
```

      And replace it with this section of code:

```
        SELECT DTree.Name, DTree.ModifyDate, DTree.ModifiedBy
        FROM DTree
        WHERE DTree.ModifyDate > %7
        AND DTree.SubType = 144
        AND %8
        ~3
```

   c. In the **Templates** section, edit the "~1" template.

      Remove this section of code:

```
        SELECT Count(AuditID) as Count
```

      And replace it with this section of code:

```
        SELECT Count(Name) as Count
```

   d. In the **Templates** section, edit the "~2" template.

      Remove this section of code:

```
    SELECT ROW_NUMBER() OVER(ORDER BY %2 %3) AS RowNumber,
AuditID,  AuditDate, Name, SubType, PerformerID
```

And replace it with this section of code:

```
    SELECT ROW_NUMBER() OVER(ORDER BY %2 %3) AS RowNumber,
Name, ModifyDate, ModifiedBy
```

e. Click **Save and Exit**.

9. Edit the WebReport you created in Step 1, "WebRep CPDS01":

   a. From the **Functions** menu of the "WebRep CPWR01" WebReport, click **Edit Reportview**.

   b. In the **Edit WebReport** page, scroll to the bottom. Edit the bottom section so that it reads:

```
    [LL_WEBREPORT_INSERTJSON @TABLEREPORT
    COLLECTIONPROCESSING:"datasource"
    EXCLUDECOLUMNS:'["RowNumber"]'
    FORMATCOLUMNNAMES:ModifiedBy:"Modified
By":ModifyDate:"Last Date Modified"
    FORMATCOLUMNS:SubType:ModifiedBy:"[LL_REPTAG=ModifiedBy
USERINFO:FULLNAME /]":ModifyDate:"[LL_REPTAG=ModifyDate
DATE:LONG /]" /]
```

   c. Click **Add Version**.

10. Edit the WebReport you created in Step 4, "WebRep CPDS02". From the **Functions** menu of the "WebRep CPDS02" WebReport, click **Edit Reportview**.

11. In the **Edit WebReport** page, edit the content so that it reads:

```
[/* Title: Blank Reportview */]

<script>
    csui.require(['csui/lib/marionette', 'csui/utils/contexts/
page/page.context', 'webreports/controls/table.report/
table.report.view'], function (Marionette, PageContext,
TableReportView) {

         var contentRegion = new Marionette.Region({el:
'#content'}),
             pageContext   = new PageContext(),
             tableReportView,
             options;

         options = {
            context: pageContext,
            data: {
                id: 463411,
                title: 'Documents Modified in the Past Week',
                header: false,
                titleBarIcon: 'title-assignments',
                columnsWithSearch: 'name',
                sortBy: 'Name',
```

```
                sortOrder: 'desc'
            }
        };

        tableReportView = new TableReportView(options);
        contentRegion.show(tableReportView);
        pageContext.fetch();

    });
</script>

<div id="content"></div>

[LL_WEBREPORT_STARTROW /]
[LL_WEBREPORT_ENDROW /]
```

12. Before you save your changes, edit the line:

    ```
    id: 463411,
    ```

    to put in the ID that you noted from the first WebReport in Step 1.

13. While in Smart View, browse to the test folder in which you saved the perspective. For the purposes of this example, browse to "DatSrc TableReport Perspective".

3.3 Configuring the Widget Carousel Widget

The Widget Carousel widget allows you to display up to ten different slides in the space of one Smart View tile. You can populate each slide with a child widget from the Content Intelligence widget group.

Widget Carousel Widget Configuration Parameters

Name	Description
Widget Size	Mandatory. Select the size for the Widget Carousel widget in the Flow layout: Single width, Double width, or Full Page. Default = Double width.
Show Header	Optional. Select whether or not the tile will include a header that contains the title and title icon. If set to `True`, the header will appear. If set to `False`, there will be no header. Default = `False` **Notes** • If the tile header is enabled and the child widget already has a header, then *both* headers will appear. • In most cases, you would only enable the Widget Carousel header if all the included child widgets did *not* have a header.

Title	Optional. Enter the title for the Widget Carousel. If the header is enabled, the title will appear on the header. Default title = "Widget Carousel"
Icon class	Optional. Enter the CSS class for the icon that will appear beside the **Title** in the **Header**. For example: `support/csui/themes/carbonfiber/icons.css` contains icons such as `title-assignments`, `title-customers`, `title-favourites`, `title-opportunities`, `title-recentlyaccessed`, `title-activityfeed`, `title-customviewsearch`. Default = "title-webreports" (the WebReports icon ) **Note:** You can specify a custom icon within the WebReport template by adding a CSS selector that uses `background-image` and then referencing the selector in this option.
Behavior	Set the options that control the behavior of the Widget Carousel. **Cycle automatically** Optional. Select whether or not the Widget Carousel will rotate through the slides. If set to `True`, the carousel will automatically rotate through the slides. If set to `False`, the carousel will only advance to the next slide if the user explicitly clicks on the navigation controls. Default = `True` **Interval** Optional. If the Widget Carousel is set to cycle automatically, set the amount of time, in milliseconds, each slide will appear before advancing to the next one. Default = 5000 (ms) **Pause on hover** Optional. If the Widget Carousel is set to cycle, and this option is set to `True`, the carousel will stop cycling if the user explicitly points to the current slide. If set to *False*, the carousel continues to cycle, regardless of where the user points the cursor. Default = `True` **Loop** Optional. If the Widget Carousel is set to cycle automatically and Loop is set to `True`, the carousel will advance through all the slides in an endless loop. If the Widget Carousel is set to cycle automatically and Loop is set to *False*, the carousel will only show each slide once. Default = `True`

Child widgets	Mandatory. Add and configure up to ten slides in the order in which you want them to appear in the carousel. You can populate each slide by dragging a child widget from the **Content Intelligence** widget group to the **Child Widget Drop Area**. For more information about how to configure child widgets from the **Content Intelligence** widget group, see the following topics: • "Configuring the Nodes List WebReport Widget" on page 65 • "Configuring the HTML WebReport Widget" on page 67 • "Configuring the Visual Count Widget" on page 85 **Notes** • You cannot nest a Widget Carousel within another Widget Carousel. • The total loading time of the carousel is affected by the loading time of each child widget on a slide. In the Smart View, a perspective that includes a Widget Carousel widget finishes loading only *after the first slide fully loads*. After the first slide loads, the carousel advancement controls activate *only after the remaining slides finish loading*. For best performance, choose quick-loading slides, particularly the first one, to include in the Widget Carousel.

3.3.1 To Configure the Widget Carousel Widget

To configure the Widget Carousel widget:

1. Create up to ten Content Intelligence widgets that you want to include on a slide in the Widget Carousel. For information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.
2. To open the Perspective Manager, navigate to the **Content Server Administration** > **ActiveView Administration** page, and click the **Open the Perspective Manager** link.
3. In the Perspective Manager, create a new perspective or edit an existing perspective. For more information about how to create a new perspective or how to edit a perspective, see the *Online Help* available in the Perspective Manager.
4. On the **Configure** tab, do the following:
   a. In the **Widget Library** pane, click to expand the **Content Intelligence** widget group and drag the **Widget Carousel** widget to the page.

      Note: Each Perspective can only include *one* Widget Carousel widget.

   b. To configure the **Widget Carousel widget**, do the following:

      i. In the work area, click the **Widget Carousel widget**.

ii. In the **Options** pane, configure the settings for the **Widget Carousel widget** as described in "Widget Carousel Widget Configuration Parameters" on page 82.

c. To add and configure a child widget as a slide in the Widget Carousel, do the following:

i. To add a child widget as a slide in the **Widget Carousel widget**, drag a widget from the **Content Intelligence** widget group to the **Widget Carousel widget**, into the **Child Widget Drop Area**.

Notes

- You cannot nest Widget Carousel widgets.
- Each child widget must call a unique WebReports node ID. You cannot reuse or repeat the same WebReport on more than one slide in the Widget Carousel.
- As you drag more widgets to the Widget Carousel, the completion bar will indicate how much room remains for additional child widgets, up to a maximum of ten.

ii. In the **Options** pane, expand the **Child widgets** section and click the child widget that you want to configure.

iii. In the updated **Options** pane for the child widget, configure the settings as appropriate for the type of Content Intelligence widget selected.

- "Configuring the Nodes List WebReport Widget" on page 65
- "Configuring the HTML WebReport Widget" on page 67
- "Configuring the Visual Count Widget" on page 85

d. If needed, repeat Step 4.c for up to a maximum of ten child widgets.

5. Navigate to the location where the new perspective is applied to verify the Widget Carousel widget output.

3.4 Configuring the Visual Count Widget

The Visual Count widget helps visualize quantifiable data as a chart on a tile in the Smart View. This widget is useful when you have a data source that has columns with values that you would like to compare.

Each Visual Count widget depends on a WebReports object to format its data, which can be any valid WebReports data source. For more information about how to create these prerequisites, see "Examples and Walkthroughs of How to Use the Visual Count Widget" on page 91.

You can use the Perspective Manager to configure the initial appearance of the chart in the Smart View, but, depending on the widget configuration, the end user may be able to use the controls in the Smart View to sort and filter the chart.

Smart View end users can change the filtering of the Visual Count widget in the following ways:

- *From the Tile view*, if an administrator uses the Perspective Manager to set **Enable Tile Filtering Controls** option to `True`.
- *From the Expanded view*, if an administrator uses the Perspective Manager to set **Enable Expanded View** option to `True`. The expanded view also allows filtering by multiple columns.

Visual Count Widget Configuration Parameters

Name	Description
Widget Size	Mandatory. Select the size for the Visual Count widget in the Flow layout: Single width or Double width. Default = Double width.
Title	Optional. Enter the title for the widget. The title appears in the tile header. Default = "Visual Count"
Chart Type	Optional. Select the type of chart appropriate for your data. Options include *Bar* chart, *Donut* chart, and *Pie* chart. Default = Bar chart
Source WebReport	Mandatory. Browse for the WebReport that retrieves the data that you want to visualize. **Note:** You can use the following reportviews that have been specifically created to use with this widget and edit them as required. • widget_visual_count_grouping_on_client • widget_visual_count_grouping_in_webreport • widget_visual_count_grouping_in_data_source_simple • widget_visual_count_grouping_in_datasource For more information about the `@NODESTABLEFIELDS` directive, and other directives that specify WebReports Smart UI reports output, see the INSERTJSON tag.

Active Column	Mandatory. Enter the name of the column from the data source that provides the matching values to be counted. Default = The name of the first column in the data source. **Notes** • When you enter the column name, you must be careful of the case sensitivity if your data source is a LiveReport that queries a case-sensitive database. • Avoid choosing an **Active Column** that returns unique values because the data will not be meaningful on a count-based chart. Ideally, this widget works best for data with columns containing no more than 20 distinct values. • Include fewer than ten columns to make the chart more user-friendly. More than ten columns are more complex to configure and may potentially have a performance cost. • You can include and exclude columns by editing the reportview used for the WebReports object associated with this widget. • If **Enable Tile Filtering Controls** is set to `True`, a Smart View end user can select a different **Active Column** value for the chart. For more information, see ??? on page 88.
Values As Percentages	Choose whether to show the values as percentages of the total count. If `False`, the values will appear as the actual count instead. Default = False
Group After	Optional. Choose the number, from one to twenty, as the threshold for the maximum number of discrete values whose total counts will appear on the chart. Values that exceed this threshold are grouped as "Other" and will show the combined total. Options include values from one to twenty and "Use Chart Default", which will use the default number appropriate for the chart type selected. • Bar chart default = 15 • Pie chart default = 5 • Donut chart default = 5
Sort By	Optional. Choose the sorting criteria for the chart. Options include the following: • **Count** – Sort numerically by the total count of similar values. • **Active Column** – Sort alphanumerically, using the actual values from the **Active Column** selection described in ??? on page 87. Default = Count
Sort Direction	Optional. Choose whether the criteria specified in the ??? on page 87 setting are sorted in ascending or descending order. Default = Descending

Chart Options	Optional. Choose whether Smart View end users can access the filter menu on the Tile view of the chart to change how they want to filter the chart data. If set to `True`, the end user can see the **Configure** icon on the Tile view. If set to `False`, then the end user cannot see the **Configure** icon and cannot filter the chart in the Tile view. Default = False **Note:** This setting only affects the end user ability to filter the chart in the Tile view, *not* the Expanded view.
Expanded View	Optional. Choose whether Smart View end users can expand the Tile view of the chart to see more chart details as well as change how they want to filter the chart data. Default = False
Button WebReports	Optionally, configure custom buttons for the Expanded view. Each button can pass in fixed values and then launch the associated WebReport. **Maximum Items to Process** Enter the maximum number of rows in the data set that the Button WebReport will process. If the filtered data set exceeds this limit, the WebReport launched by the button will return an error. Default = 5000 rows **Note:** There is a hard limit of 50,000 rows. If you try to set a value higher than 50,000, you will get an error.
Custom Buttons	Optionally, add one or more custom buttons to the footer of the Expanded view. Configure each button to launch the selected WebReport in a new window, optionally passing in preset parameters. • **Button WebReport** – Optional. Expanded view only. Browse for the WebReport that will open when you click the button. Default = If left blank, no custom button or footer appears. • **Button Label** – Optional. Expanded view only. Enter a custom label that will override the default label for the custom button. Default = "Launch WebReport" • **Button Tooltip** – Optional. Expanded view only. Enter a custom tool tip that will override the default tool tip for the custom button. Default = "Launch a WebReport in a new window using the current filters"
Source Parameters	Optional. If the Source WebReport that generates the data to be visualized accepts parameters, you can specify them here. • **Parameter Name** – Optional. Name of the parameter that you want to pass to the WebReport that generates the data to be visualized. • **Parameter Value** – Optional. Value of the parameter that you want to pass to the WebReport that generates the data to be visualized.

3.4.1 To Configure the Visual Count Widget

Important
Be careful when using the Visual Count widget in a frequently loaded location, such as the landing page.

To configure the Visual Count widget:

1. Create a WebReport to generate the data that you want to visualize as a chart with the Visual Count widget. Choose from the following reportview templates that generate data compatible with the Visual Count widget:

   - widget_visual_count_grouping_on_client
   - widget_visual_count_grouping_in_webreport
   - widget_visual_count_grouping_in_data_source_simple
   - widget_visual_count_grouping_in_datasource

   For information about which reportview template to use, see "Choosing a Reportview Template" on page 90. For information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

   Note: You will need to open your data source and make note of the column that you want to use to group the data. You will need this label to configure the widget in Step 5.c.

2. Optional Create a WebReport for each action that you want end users to be able to perform on the results in the Expanded view of the **Visual Count** widget. You will be able to configure a custom button that end users can click to launch this WebReport.

3. On the **Content Server Administration** > **ActiveView Administration** page, click the **Open the Perspective Manager** link.

4. In the Perspective Manager, create a new perspective or edit an existing perspective. For more information about how to create a new perspective or how to edit a perspective, see the *Online Help* available in the Perspective Manager.

5. On the **Configure** tab, do the following:

   a. In the **Widget Library** pane, click to expand the **Content Intelligence** widget group and drag the **Visual Count** widget to the page.

   b. To open the configuration options, select the **Visual Count** widget on the page.

   c. In the **Options** pane, configure the settings as described in "Visual Count Widget Configuration Parameters" on page 86.

      - **To select the WebReport to be visualized**

In the **Source WebReport** box, click **Browse** and select the WebReport created in Step 1.

- **To add custom buttons to the Expanded view**
  i. Click to expand the **Custom Buttons** section.
  ii. In the **Button WebReport** box, click **Browse** and select the **Button WebReport** created in Step 2.
  iii. In the **Button Label** box, enter a custom label for the custom button.
  iv. In the **Button Tooltip** box, enter a custom tooltip for the custom button.
  v. Click **Add to Array**.
  vi. Repeat Step ii through Step v to configure a button for each **Button WebReport** that you have prepared
- **To pass in parameters to the WebReport to be visualized**
  i. Click to expand the **Source Parameters** section.
  ii. In the **Parameter Name** box, enter the name of the parameter that you want to pass to the WebReport.
  iii. In the **Parameter Value** box, enter the value of the parameter that you want to pass to the WebReport.
  iv. Click **Add to Array**.
  v. Repeat Step ii through Step iv for each parameter that you want to pass to the WebReport.

6. Browse to the location where the new perspective is applied to verify the **Visual Count** widget output.

3.4.2 Choosing a Reportview Template

This section helps you decide which reportview template you should use for the WebReport that generates the data that the Visual Count widget visualizes.

The reportviews available for the WebReport and specific to the Visual Count widget include the following:

- `widget_visual_count_grouping_in_data_source_simple`
- `widget_visual_count_grouping_on_client`
- `widget_visual_count_grouping_in_webreport`
- `widget_visual_count_grouping_in_datasource`

Figure 3-1 illustrates the factors you should consider when choosing the appropriate reportview for this WebReport.

- The intensity of the data source resource query – Light, moderate, or heavy.
- The type of data source – The **widget_visual_count_grouping_in_data_source_simple** reportview works with

a simple tab-separated text file. The **widget_visual_count_grouping_on_client** reportview and **widget_visual_count_grouping_in_webreport** reportview work with any data source. While the **widget_visual_count_grouping_in_datasource** reportview currently only works with LiveReports.

- Whether or not you want to allow end users to use client browser-based controls to filter the data in the resulting chart.
- The frequency of access for the page where the Visual Count widget resides.

Figure 3-1: Type of Database Query Determines Which Reportview to Use

Tip: In any data source, to see which columns are available to use as an Active Column, click the source WebReport to run it and view the JSON output.

Important

- Be careful of using the Visual Count widget in a frequently loaded location, such as on the landing page. Frequent loading of even a small data set may result in a significant impact on performance.
- If you do use the Visual Count widget on the frequently loaded page, such as the landing page, make sure that you use a query with minimal performance impact or the your run the query using the WebReports scheduler and return the pre-grouped output such as described in "Scenario 5: Scheduled Data Chart for Very Large Data Sets (Data Layer)" on page 112.

3.4.3 Examples and Walkthroughs of How to Use the Visual Count Widget

If you plan to use the Visual Count widget, you will need to prepare one or more WebReports. This section includes scenarios that show how to configure these WebReports to retrieve the data for the chart that you want. These scenarios should be read in sequence as the examples progress from simple to more complex and each includes a walkthrough that takes you step-wise through a real-world example using Content Server data.

1. "Scenario 1: Simple Chart with Static Data" on page 92
2. "Scenario 2: Filter-enabled Chart" on page 95
3. "Scenario 3: Filter-enabled Chart for Larger Data Sets (Application Layer)" on page 100
4. "Scenario 4: Filter-enabled Chart for Very Large Data Sets (Data Layer)" on page 105
5. "Scenario 5: Scheduled Data Chart for Very Large Data Sets (Data Layer)" on page 112

Optionally, after configuring the WebReport for the main chart, you can configure additional WebReports, that the end user can launch from custom buttons in the Expanded view of the Visual Count widget. For more information about how to prepare these WebReports for the custom buttons, see "Preparing WebReports for the Custom Buttons" on page 117.

Scenario 1: Simple Chart with Static Data

This section provides examples of how to visualize data that includes values with variable totals to produce a *simple* chart that end users *do not need to filter*. Example 3-2, "Employee Count for a Retail Store" on page 92 describes a conceptual use case and the associated data. Then "Preparing to Visualize a Very Simple Chart" on page 93 shows you how to prepare the template and "Step-by-step Walkthrough for a Simple Chart with Static Data" on page 93 takes you through the process of visualizing that data using the Visual Count widget.

Example 3-2: Employee Count for a Retail Store

Consider this simple data set that lists the employee count on various days at a retail store.

Table 3-1: Employees

Weekday	Employee_Count
Sunday	3
Monday	3
Tuesday	4
Wednesday	6
Thursday	5
Friday	4
Saturday	5

In this example, the data is minimal. The data source already has a count corresponding to the grouped data. All the employees who worked on a Saturday have been grouped as "Saturday", and their total is shown as "Employee_Count". Since there are no further details for each employee, such as job title, gender, or name, there are no other criteria to count so there is no point enabling tile filtering or expanded view. In this case, we want to show a non-interactive "infographic" type of chart. The Visual Count widget allows you to plot this data as a bar chart, showing the weekday on the x axis and the employee count on the y axis. This kind of data is suitable when you need to display a graphical chart and do not require the end user to interact with the chart.

Preparing to Visualize a Very Simple Chart

To visualize a very simple data source that includes pre-grouped, static data, such as in a tab-separated value (TSV) file or a comma-separated value (CSV) file, you can use the `widget_visual_count_grouping_in_data_source_simple` reportview template.

Size	Recommended for static, pre-grouped data where the size of the data used to generate the grouped counts does not affect overall performance.
Filtering	Does not support filtering.
Data source	A very simple data source that includes pre-grouped, static data, such as in a TSV file or a CSV file.
WebReports object	Must use the `widget_visual_count_grouping_in_data_source_simple` reportview template. For an example, see "Step-by-step Walkthrough for a Simple Chart with Static Data" on page 93.
Visual Count widget configuration	No special configurations required.
Performance	• Renders static data without user interaction on the client, so it has very little additional impact on performance. • Suitable for use in a frequently-accessed location, such as a landing page.

Step-by-step Walkthrough for a Simple Chart with Static Data

To step through the process of visualizing a very simple chart with static data

1. Create a file with pre-grouped, static data:
   a. In Content Server, click **Add Item** to create a new text document.
   b. On the **Add: Text Document** page, in the **Name** box, enter a name for the file.
   c. In the **MIME Type** area, select the **Text** option, then, in the **Text** list, choose **text/tab-separated values**.
   d. In the **Text** box, type the following, pressing **TAB** to separate the columns and pressing **ENTER** to insert a line break:

```
Country         Count
Germany        1500
Canada         2000
United States  1000
```

   e. Save and close the document.
2. Create a WebReports object that will format the data as required:

   For more information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

a. In Content Server, click **Add Item** to create a new WebReports object.

b. On the **Add: WebReport** page, in the **Name** section, enter a name for the WebReport.

c. In the **Data Source**section, click **Browse Content Server** and choose the static data file created in Step 1.

d. In the **Reportview file** section, from the list, select `widget_visual_count_grouping_in_data_source_simple`.

e. On the **Source** tab, in the **Column Separator** box, clear the comma and click the **Tab** link to insert tab as the column separator.

f. Click **Add**.

3. Create a perspective to see the chart in Smart View:

a. Sign in to Perspective Manager and create a perspective.

For more information about how to create a perspective, see the *Online Help* available in Perspective Manager.

b. Configure a Visual Count widget with the following settings:

For more information about how to configure the Visual Count widget, see "To Configure the Visual Count Widget" on page 89.

Name	Value
Widget Size	If using the Flow layout, choose either Single width or Double width.
Title	Employees by Country
Chart Type	Bar
Source WebReport	<*YourWebReportID*> as created in Step 2.
Active Column	Country  **Notes** • Make sure to use the correct case because Content Server folder columns are case-sensitive. • Additional columns may be available depending on which Content Server modules are installed. • Many columns will contain unique values, such as Name, ID, and so on, and will not be useful for a chart that counts items with similar values.
Show Values as Percentages	False
Group After	Use chart default
Sort By	Count
Sort Direction	Descending

Enable Tile Filtering Controls	False
Enable Expanded View	False

c. Sign in to Smart View and verify that the Visual Count widget in the perspective produces the bar chart as follows:

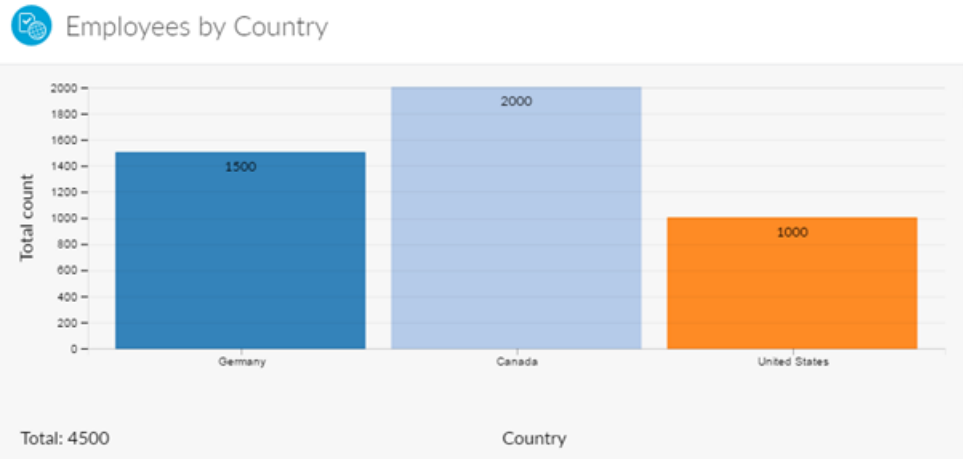

Figure 3-2: Simple Bar Chart

Scenario 2: Filter-enabled Chart

This section explores a scenario where you want to produce an *interactive* chart where the end user *can apply filters* to highlight different aspects of the data, as described in Example 3-3, "Detailed Employee Information for a Retail Store" on page 95. Then "Preparing to Visualize a Basic, Filter-enabled Chart" on page 97shows you how to prepare the template and "Step-by-step Walkthrough for a Basic, Filter-enabled Chart" on page 98takes you through the process of visualizing that data using the Visual Count widget.

Example 3-3: Detailed Employee Information for a Retail Store

Consider these first few rows of a larger data set that lists several kinds of data for each employee in a local company.

Table 3-2: EmployeeDetails

EmployeeID	Region	Gender	Function
12345	Eastbury	Male	Human Resources
12346	Midtown	Male	Sales

EmployeeID	Region	Gender	Function
12347	Downtown	Female	Sales
12348	Northglen	Female	Finance
12349	Hillside	Male	Engineering
12350	Westfield		Sales
12351			

In this example, the count is derived by grouping whichever values are the same in the column of data that is of interest, the Active Column.

If the Active Column is “Country”, we group all the rows with the same country value and then count those rows with the following results:

- England = 3
- France = 1
- USA = 3

If the Active Column is “Gender”, we need to regroup and recount with the following results:

- Male = 3
- Female = 3
- No value = 1

Notes

- If the data source has missing values, it is important to exclude them from the resulting chart to avoid misleading information. We group them as “No value”.
- The sum of each value count, including the count of missing values, if any, should be the same as the total number of rows in the data source.
- This data source includes more information than we can easily display on a bar chart or a pie chart. The more data that you retrieve, the more it affects performance, so it makes sense to only bring back the columns needed for the chart and to exclude the columns that are not needed.
- By enabling end-user filtering, the end user can choose an Active Column, so that the same data can produce several different charts. It also allows the end user to perform more complex analyses by combining filters. For example, “Total count of Sales employees in Europe, by gender”.

Preparing to Visualize a Basic, Filter-enabled Chart

If you have a small data set from a standard WebReports data source query that is minimally intensive, you can use the `widget_visual_count_grouping_on_client` reportview template. This reportview relies on the javascript engine of the client browser to filter the entire data set.

Size	Recommended for small data sets where the results can quickly load to the client browser.
Filtering	Allows filtering by the end users using the client browser.
Data source	Any standard WebReports data source.
WebReports object	Must use the `widget_visual_count_grouping_on_client` reportview template.
Visual Count widget configuration	• No special configurations required. • Supports a Visual Count widget configuration with **Enable Tile Filtering Controls** = `True` or **Enable Expanded View** = `True` so that end users can use the client browser to group and filter the data.
Performance	• After the Visual Count widget finishes loading, end-user filtering will be fast and responsive because no additional requests are sent back to the server. • OpenText does not recommend using for a chart in a frequently loaded location, such as on the landing page. Frequent loading of even a small data set may result in a significant impact on performance. • Not recommended for medium- to large-sized data sets.

Considerations for a Basic, Filter-enabled Chart

- **Performance Considerations**
  - In this scenario, the entire data set is returned to the browser. The browser filters the data using the client-side Javascript layer.
  - The data set size is affected by both the number of columns and the amount of data it contains. For example, a report with 10 columns that only contains integers will process quickly. However, a report with only two columns will process more slowly if that report includes a cell containing the lengthy description field.
  - It is a good practice to tune your LiveReport to only return the data that you need. For information about how to optimize a LiveReport, see *OpenText Content Server - LiveReports (LLESREP-UGD)*.
- **Advantages**
  - No requests are made to the server when the end user uses the browser to change the Active Column or to choose other chart options.
  - Performance is quick when the data set is small.
- **Disadvantages**

Performance may degrade if the data set is large. The browser will use more memory to manipulate the data as the data set size increases. To address this concern, see "Step-by-step Walkthrough for a Basic, Filter-enabled Chart" on page 98.

Step-by-step Walkthrough for a Basic, Filter-enabled Chart

For this example we will use a Content Server folder as a data-source. To support filtering by the end user, perform the following procedures:

To step through the process of visualizing a basic, filter-enabled chart

1. Create a WebReports object that will format the data as required:

   For more information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

   a. In Content Server, click **Add Item** to create a new WebReports object.
   b. On the **Add WebReport** page, in the **Name** section, enter a name for the WebReport.
   c. In the **Data Source**section, click **Browse Content Server** and choose a folder that contains several different document types.
   d. In the **Reportview file** section, from the list, select `widget_visual_count_grouping_on_client.`.
   e. Click **Add**.
   f. Optional Edit the reportview for WebReport created in Step 1:

      i. Optional Add an `INCLUDECOLUMNS` definition:

      ```
      INCLUDECOLUMNS:"{'MimeType','FileType','SubType'}"
      ```

      This definition is optional, but helps to improve the overall user experience for the end user:

      - Reduces the overall size of the JSON data returned by removing unnecessary information.
      - Removes irrelevant options from the Active Column list in the Tile filter controls and in the Expanded view.

      Note: Unless you explicitly use `INCLUDECOLUMNS` or `EXCLUDECOLUMNS`, the Tile filter controls will list all available columns.

      ii. Click **Add Version**.

2. Create a perspective to see the chart in Smart View:

   a. Sign in to Perspective Manager and create a perspective.

   For more information about how to create a perspective, see the *Online Help* available in Perspective Manager.

b. Configure a Visual Count widget with the following settings:

For more information about how to configure the Visual Count widget, see "To Configure the Visual Count Widget" on page 89.

Name	Value
Widget Size	If using the Flow layout, choose either Single width or Double width.
Title	Documents by type
Chart Type	Pie
Source WebReport	*<YourWebReportID>* as created in Step 1.
Active Column	FileType **Notes** • Make sure to use the correct case because Content Server folder columns are case-sensitive. • Additional columns may be available depending on which Content Server modules are installed. • Many columns will contain unique values, such as Name, ID, and so on, and will not be useful for a chart that counts items with similar values.
Show Values as Percentages	False
Group After	Use chart default
Sort By	Count
Sort Direction	Descending
Enable Tile Filtering Controls	True
Enable Expanded View	True

c. Sign in to Smart View and verify that the Visual Count widget in the Perspective produces the pie chart as follows:

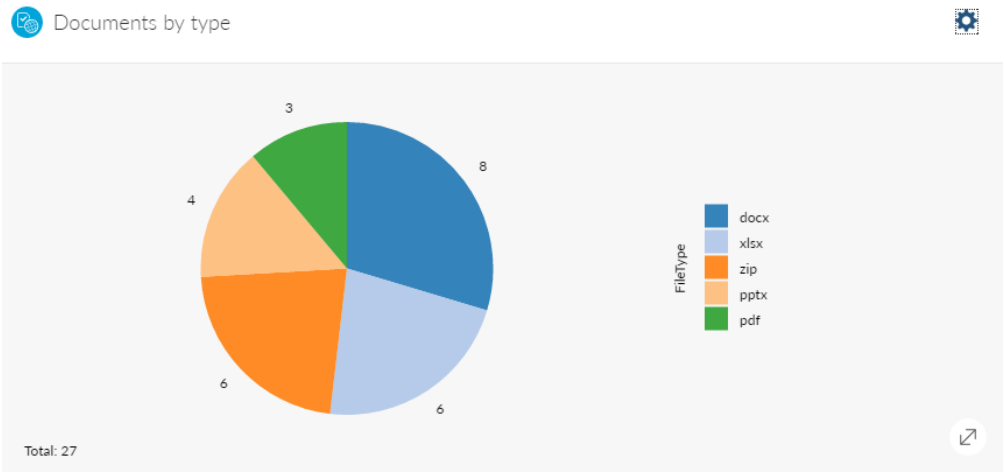

Figure 3-3: Basic Filter-enabled Pie Chart

Scenario 3: Filter-enabled Chart for Larger Data Sets (Application Layer)

This section provides examples of how to limit the data retrieved from a potentially large data set to produce a chart that end users can filter to get further information. Example 3-4, "Employee Information Data for Acme National" on page 101 describes a conceptual use case for employee information from a national company. Then it illustrates an example using file data from Content Server, while steps you through the process of visualizing the Content Server file data in a filter-enabled chart using the Visual Count widget. Since the chart is filter-enabled, we can optionally enable the expanded view for the chart to allow facet filtering. For more information about how to use facet filtering, see *OpenText Content Server - Documents and Text Documents (LLESWBD-UGD)*.

Next, "Preparing to Visualize a Filter-enabled Chart for Larger Data Sets" on page 102 shows you how to prepare the template and "Step-by-step Walkthrough for a Filter-enabled Chart for Larger Data Sets" on page 103 takes you through the process of visualizing that data using the Visual Count widget.

To produce a chart that performs well with larger datasets, we need to limit the data returned from the server to only the subset of data required to visualize the chart.

You can limit the data by editing the data source or the SQL query to return only the columns that you need. However, this is not always possible. The user creating the WebReport may not have the SQL skills or permission to edit the LiveReport data source. If you are comfortable working with WebReports, it may be more convenient to edit the data at the application layer, by using conditional logic and sub-tags in the reportview. You can also exclude columns by using directives, as illustrated in the commented section of the `widget_visual_count_grouping_in_webreport` reportview template. Rather than return all the data to the client browser in a single request, the application layer can provide the Visual Count widget with the

minimum data required to create the chart. In this case, the application layer need only return the data relating to the active column that contains the value counts that we want to visualize. To show the chart for a different active column, or with different filter criteria, you can make a new request to the server for fresh data that contains the information you need.

Example 3-4: Employee Information Data for Acme National

Consider these first few rows and columns of a much larger data set that lists several kinds of data for each employee in a national company.

Table 3-3: EmployeeInfo

EmployeeID	State	Gender	Function	City	Years	Supervisor	...
12345	Nebraska	Male	Engineering	Omaha	5	Smith	...
12346	Texas	Male	Sales	Austin	6	Jones	...
12347	New Jersey	Female	Finance	Newark	12	White	...
12348	California	Female	Sales	Irvine	4	Lee	...
12349	Ohio	Male	Engineering	Toledo	5	Wilson	...
12350	Arizona		Engineering	Phoenix	7	Roberts	...
12351	Wisconsin	Female	Human Resources	Kenosha	1	Hansen	...

In this example, the count is derived by grouping whichever values are the same in the column of data that is of interest, the Active Column.

If the Active Column is “Function”, we group all the rows with the same function value and then count those rows with the following results:

- Sales = 2
- Engineering = 3
- Finance = 1
- Human Resources = 1

If the Active Column is “Gender”, we need to regroup and recount with the following results:

- Male = 3
- Female = 3
- No value = 1

Note: If the data source has missing values, it is important NOT to exclude them from the resulting chart to avoid misleading information. Even though they have no value, they still need to be counted. For example, an employee who has not specified their gender is still an employee and must still be counted. In the Visual Count widget, these "unknown" values are grouped as "No value".

Preparing to Visualize a Filter-enabled Chart for Larger Data Sets

If you have a larger data set from a standard WebReports data source that requires a moderately-intensive data source query, and if you cannot group and filter the data in the data source, you can use the `widget_visual_count_grouping_in_webreport` reportview template. This reportview *groups and filters the data in the WebReports object.*

Size	Useful for larger data sets that take longer to load to the client browser.
Filtering	Requires filtering at the WebReports object.
Data source	Any standard WebReports data source.
WebReports object	Must use the `widget_visual_count_grouping_in_webreport` reportview template.
Visual Count widget configuration	• No special configurations required. • Supports a Visual Count widget configuration with **Enable Tile Filtering Controls** = `True` or **Enable Expanded View** = `True` so that end users can use the client browser to group and filter the data.
Performance	• Improved performance between client and server because it does not return all data rows to the client browser. • Performance dependent on data transfer between the database and Content Server before the WebReports object can group and filter the data. • OpenText does not recommend using for a chart in a frequently loaded location, such as on the landing page. Frequent loading of even a small data set may result in a significant impact on performance.

Performance Considerations for a Filter-enabled Chart for Larger Data Sets

Note: This approach suits larger data sets when you expect a moderately large number of results and do not want to retrieve all the data at once.

- In this scenario, only a subset of the data set is returned to the browser. The server-side WebReports application filters the data.
- Each time an end user changes the Active Column or changes the filtering settings, the widget passes parameters to the WebReport, which then makes a new server request that returns a new data set.

- It is a good practice to tune your LiveReport to only return the data columns that you need.

Step-by-step Walkthrough for a Filter-enabled Chart for Larger Data Sets

To step through the process of visualizing a filter-enabled chart with a larger data set

1. Create a WebReports object that will format the data as required:

   For more information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

   a. In Content Server, click **Add Item** to create a new WebReports object.
   b. On the **Add WebReport** page, in the **Name** section, enter a name for the WebReport.
   c. In the **Data Source**section, click **Browse Content Server** and choose a folder that contains several different document types.
   d. In the **Reportview file** section, from the list, select `widget_visual_count_grouping_in_webreport`.
   e. Click **Add**.
   f. Optional Edit the reportview for the WebReport created in Step 1.

      i. Optional Add an `INCLUDECOLUMNS` definition:

      ```
      INCLUDECOLUMNS:"{'MimeType','FileType','SubType'}"
      ```

      This definition is optional, but helps to improve the overall user experience for the end user:

      - Reduces the overall size of the JSON data returned by removing unnecessary information.
      - Removes irrelevant options from the Active Column list in the Tile filter controls and in the Expanded view.

      Note: Unless you explicitly use `INCLUDECOLUMNS` or `EXCLUDECOLUMNS`, the Tile filter controls will list all available columns.

      ii. Optional Add a `FORMATCOLUMNS` definition:

      ```
      FORMATCOLUMNS:SubType:"[LL_REPTAG=SubType
      LABEL:SubType /]"
      ```

      This definition is optional. It will take all the values in the SubType column and give them a friendly label. For example, “0” becomes “Folder” and “144” becomes “Document”.

      iii. Click **Add Version**.

2. Create a perspective to see the chart in Smart View:

a. Sign in to Perspective Manager and create a perspective.

   For more information about how to create a perspective, see the *Online Help* available in Perspective Manager.

b. Configure a Visual Count widget with the following settings:

   For more information about how to configure the Visual Count widget, see "To Configure the Visual Count Widget" on page 89.

Name	Value
Widget Size	If using the Flow layout, choose either Single width or Double width.
Title	Test
Chart Type	Bar
Source WebReport	<*YourWebReportID*> as created in Step Step 1.
Active Column	MimeType  **Notes** • Make sure to use the correct case because Content Server folder columns are case-sensitive. • Additional columns may be available depending on which Content Server modules are installed. • Many columns will contain unique values, such as Name, ID, and so on, and will not be useful for a chart that counts items with similar values.
Show Values as Percentages	False
Group After	Use chart default
Sort By	Count
Sort Direction	Descending
Enable Tile Filtering Controls	True
Enable Expanded View	True

c. Sign in to Smart View and verify that the Visual Count widget in the perspective produces the bar chart.

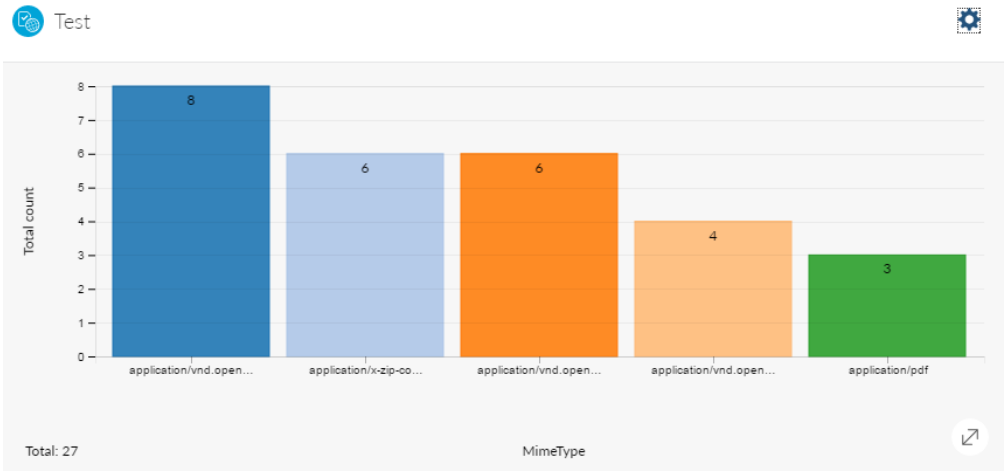

Figure 3-4: Filter-enabled Bar Chart (Larger Data Set)

Scenario 4: Filter-enabled Chart for Very Large Data Sets (Data Layer)

This section provides examples of how to limit the data retrieved from a very large data set to produce a chart that end users can filter to get further information. Example 3-5, “Employee Information Data for GlobalMega Corp.” on page 105describes a conceptual use case for employee information from a global company. Next, “Preparing to Visualize a Filter-enabled Chart from a Very Large Data Set” on page 106 shows you how to prepare the template and “Step-by-step Walkthrough for a Filter-enabled Chart from a Very Large Data Set” on page 107 takes you through the process of visualizing that data using the Visual Count widget.

To produce a chart that allows better performance with larger datasets, where the filtering task is performed at the data layer. The SQL query of the LiveReport groups all the values. To enable end users to change the Active Column and otherwise filter the data in the expanded view, you can design the LiveReport to use input parameters to make dynamic queries.

Example 3-5: Employee Information Data for GlobalMega Corp.

Consider these first few rows and columns of a much larger data set that lists several kinds of data for each employee in a huge global corporation. If we want a chart that shows the number of employees in each country, we can create a SQL query that groups by country and counts the rows for each. By changing the parameters, we can achieve different results to explore the data in a different way, for example, by choosing to group by Gender instead of Country.

Table 3-4: EmployeeData

EmployeeID	Country	Region	Gender	Function	City
12345	USA	Nebraska	Male	Engineering	Omaha
12346	USA	Texas	Male	Sales	Austin
12347	USA	New Jersey	Female	Finance	Newark
12348	France	Normandie	Female	Sales	Rouen
12349	France	Centre	Male	Engineering	Paris
12350	UK	Berkshire		Engineering	Reading
12351	UK	Hampshire	Female	Human Resources	Southampton
...	...	...	...	...	...

Preparing to Visualize a Filter-enabled Chart from a Very Large Data Set

If you have a large data set, such as from a LiveReport data source, that requires a heavily-intensive database query, you can group and filter the data in the LiveReport data source and then use the `widget_visual_count_grouping_in_datasource_simple` reportview template for the WebReports object.

Note: The `widget_visual_count_grouping_in_datasource` reportview template only works in conjunction with a LiveReport data source since it depends on the LiveReport ability to group counts of a specific column in the data source.

Size	Useful for larger data sets that would take too long to return from the database in full.
Filtering	Requires filtering at the LiveReport data source.
Data source	• Currently only works with a LiveReport data source. • Must group and configure the data in the LiveReport.
WebReports object	Must use the `widget_visual_count_grouping_in_datasource` reportview template.
Visual Count widget configuration	• No special configurations required. • Recommended Visual Count widget configuration should specify **Enable Expanded View** = `False`, as interactive filtering is not possible with this data.

Performance	• Improved performance between client and server because it does not return all data rows to the client browser. The rows are grouped and filtered in the data source. • OpenText does not recommend using for a chart in a frequently loaded location, such as on the landing page. Frequent loading of even a small data set may result in a significant impact on performance.

Performance Considerations for a Filter-enabled Chart from a Very Large Dataset

Note: This approach is most suitable for situations where the amount of data makes it more appropriate for the database engine to perform the task of filtering and grouping the data.

- In this scenario, only a subset of the results are returned to the browser. The server-side SQL code in the LiveReport filters the results at the data layer.
- It is a good practice to tune your LiveReport to only return the data that you need.

Important

Each time an end user loads the page, the widget makes a server request and a database request. Caution should be applied – if multiple users are likely to be accessing the chart and interacting with filter parameters, the load on the database could be significant. Before using this approach you can consider either of the following strategies:

- Return a smaller, pre-filtered subset of the data and using the approaches described in "Scenario 2: Filter-enabled Chart" on page 95 and "Scenario 3: Filter-enabled Chart for Larger Data Sets (Application Layer)" on page 100.
- Schedule the data to be fetched from the database periodically and use the cached results to generate a chart, as described in "Scenario 5: Scheduled Data Chart for Very Large Data Sets (Data Layer)" on page 112.

Step-by-step Walkthrough for a Filter-enabled Chart from a Very Large Data Set

Note: These examples have been tested for MSSQL and Oracle databases. If you are using a different type of database, the syntax may differ.

To step through the process of visualizing a very simple chart with static data

1. Create a filtered LiveReport data source:

   For more information about how to create a LiveReport, see *OpenText Content Server - LiveReports (LLESREP-UGD)*.

   a. In Content Server, click **Add Item** to create a new LiveReport.

b. Edit the report and provide a **Title** and a **Record Limit**.

c. In the **Inputs** section, add the following values:

#	Type	Prompt	URL Parameter
1	InsertString	ActiveColumn	active_column
2	InsertString	Sort Column	sort_by
3	InsertString	Sort Order	sort_order
4	SQLList	AuditID	AuditID
5	SQLList	SubType	SubType
6	SQLList	PerformerID	PerformerID

- Values 1-3 are mandatory and ensure the Visual Count widget passes in the correct active column and sort details to group and sort the data when the user changes values in the widget controls.
- Values 4-6 are optional and are used by the filtering feature in the Expanded view. For every column that you include in your report, you need to include an input in this section.

Note: The URL Parameter name must match the data source column name. The URL Parameter type must be SQLList so that the widget can send in one or more filters for each column.

d. Edit the LiveReport to add the following SQL code to the SQL section:

```
SELECT %1, Count(%1) as Count
FROM DAuditNew
WHERE AuditDate > %7
AND AuditID IS NOT NULL
AND SubType IS NOT NULL
AND PerformerID IS NOT NULL
~1
~2
~3
Group By %1
ORDER BY %2 %3
```

After the parameters and conditional templates have been substituted, this should build a SQL statement similar to the following:

```
SELECT AuditID, Count(AuditID) as Count
FROM DAuditNew
WHERE AuditDate > { ts '2017-01-07 23:59:59' }
AND AuditID IS NOT NULL
AND SubType IS NOT NULL
AND PerformerID IS NOT NULL
AND AuditID IN ( 'Create','AddVersion' )
AND SubType IN ( 0,144 )
AND PerformerID IN ( 1000 )
Group By AuditID ORDER BY Count Desc
```

e. In the Parameters section, add the following settings:

#	Type	Options
%1	User Input	Input #1
%2	User Input	Input #2
%3	User Input	Input #3
%4	User Input	Input #4
%5	User Input	Input #5
%6	User Input	Input #6
%7	Last Week-End	

- Parameter %1 maps to Input 1 and ensures the current Active Column is used for all %1 values in the SQL.
- Parameter %2 maps to Input 2 and ensures the current Sort Column is used for all %2 values in the SQL.
- Parameter %3 maps to Input 3 and ensures the current Sort Direction is used for all %3 values in the SQL.
- Parameter %4 maps to Input 4 and ensures the current list of filter values for the AuditStr column is used for all %4 values in the ~1 template.
- Parameter %5 maps to Input 5 and ensures the current list of filter values for the SubType column is used for all %5 values in the ~2 template.
- Parameter %6 maps to Input 6 and ensures the current list of filter values for the PerformerID column is used for all %6 values in the ~3 template.
- Parameter %7 will ensure only audit records from the end of the last week will be shown. This will reduce the performance impact of the report.

f. In the Templates section, ensure that the **Auto-Where** check box and **Auto-Comma** check boxes are clear and then add the following values:

#	SQL Source	Include IF
~1	AND AuditID IN%4	No inputs set to flag
~2	AND SubType IN%5	No inputs set to flag
~3	AND PerformerID IN%6	No inputs set to flag

Note: Ensure that the Options Flag Value is blank for all three values.

These three templates are used for the three filters for the AuditID, SubType, and PerformerID columns. These ensure the filters are only

inserted into the SQL if a filter is applied for that column. For example, if the Visual Count widget sends in the values “0” and “144” for SubType (only show audit events for folders and documents), then the following line will be added to the SQL in place of the ~2 marker:

```
AND SubType IN (0,144)
```

g. In the **Report Format** area, select **Auto LiveReport** from the list.

h. Click **Save and Exit**.

2. Create a WebReports object that will format the data as required:

For more information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

a. In Content Server, click **Add Item** to create a new WebReports object.

b. On the **Add WebReport** page, in the **Name** section, enter a name for the WebReport.

c. In the **Data Source**section, click **Browse Content Server** and choose the LiveReport created for this example in Step 1.

d. In the **Reportview file** section, from the list, select `widget_visual_count_grouping_in_data_source`.

e. Click **Add**.

f. Edit the properties for the WebReport created in Step 2.b:

i. Click the **Destination** tab.

ii. In the **Export MIME Type** list, select **application/json**.

iii. Click **Update**.

g. Edit the reportview for WebReport created in Step 2.bwith the following changes:

i. Add an `INCLUDECOLUMNS` definition:

```
INCLUDECOLUMNS:"{'AuditID','PerformerID','SubType'}"
```

The `INCLUDECOLUMNS` definition tells the Filtered Count widget which columns are available for grouping and filtering. This list must match the columns defined for filtering in the data source.

ii. (Optional) Add a `FORMATCOLUMNS` definition:

```
FORMATCOLUMNS:AuditID:"[LL_REPTAG=AuditID
LABEL:AuditID /]":PerformerID:"[LL_REPTAG=PerformerID
USERINFO:FULLNAME /]":SubType:"[LL_REPTAG=SubType
LABEL:SubType /]"
```

This definition is optional. It will take all the values in the AuditID column, PerformerID column, and SubType column and gives them a friendly label. For example, with the SubType column, “0” becomes “Folder” and “144” becomes “Document”.

iii. (Optional) Add a `FORMATCOLUMNNAMES` definition:

```
FORMATCOLUMNNAMES:AuditID:"Event
Type":PerformerID:"[LL_REPTAG_'WEBNODE_LABEL.User'
XLATE /]"
```

This definition is optional. It gives friendly labels to the specified columns. In this example, you can give the AuditID column a hardcoded string, such as "Event Type". Or you can use any WebReports tag such as shown above where the `XLATE` tag replaces PerformerID with "User". This tag detects the user language and uses the appropriate translation.

iv. After these changes, the final entire tag definition will be:

```
[LL_WEBREPORT_INSERTJSON @FILTEREDCOUNT
ACTIVECOLUMN:"[LL_REPTAG_&active_column /]"
INCLUDECOLUMNS:"{'AuditID','PerformerID','SubType'}"
FORMATCOLUMNS:AuditID:"[LL_REPTAG=AuditID
LABEL:AuditID /]":PerformerID:"[LL_
REPTAG=PerformerID
USERINFO:FULLNAME /]":SubType:"[LL_REPTAG=SubType
LABEL:SubType /]"
FORMATCOLUMNNAMES:AuditID:"Event
Type":PerformerID:"[LL_REPTAG_'WEBNODE_LABEL.User'
XLATE /]"
FILTERBY:"[LL_REPTAG_&fc_filters /]"
GROUPBYINDATASOURCE:"TRUE"
GROUPAFTER:"[LL_REPTAG_&group_after /]"
SORTCOL:"[LL_REPTAG_&sort_by /]"
SORTDIR:"[LL_REPTAG_&sort_order /]"
COUNTFORMAT:"SI" /]
```

h. Click **Add Version**.

3. Create a perspective to see the chart in Smart View:

a. Sign in to Perspective Manager and create a perspective.

For more information about how to create a perspective, see the *Online Help* available in Perspective Manager.

b. Configure a Visual Count widget with the following settings:

For more information about how to configure the Visual Count widget, see "To Configure the Visual Count Widget" on page 89.

Name	Value
Widget Size	If using the Flow layout, choose either Single width or Double width.
Title	Audit Events for the Past Week
Chart Type	Bar
Source WebReport	<*YourWebReportID*> as created in Step 2.
Active Column	AuditID

Show Values as Percentages	False
Group After	Use chart default
Sort By	Count
Sort Direction	Descending
Enable Tile Filtering Controls	True
Enable Expanded View	True

c. Sign in to the Smart View and verify that the Visual Count widget in the Perspective produces the bar chart.

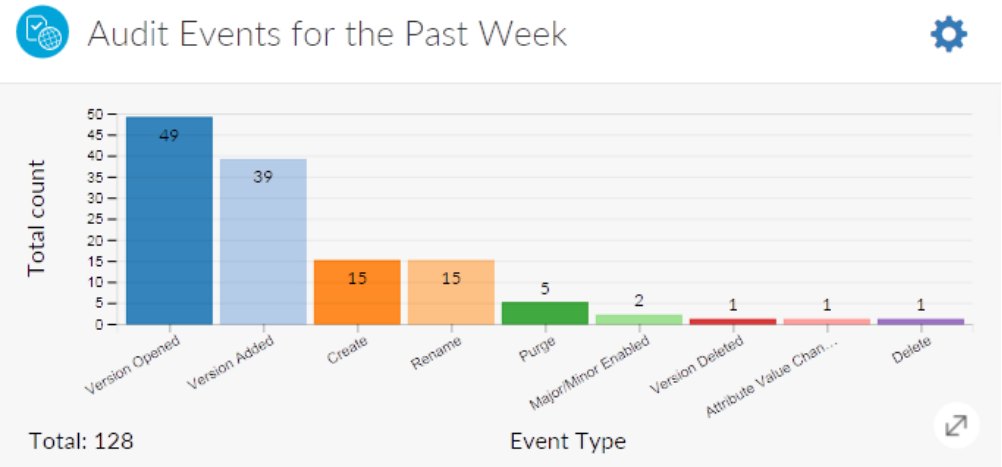

Figure 3-5: Filter-enabled Bar Chart (Very Large Data Set)

Scenario 5: Scheduled Data Chart for Very Large Data Sets (Data Layer)

This section provides examples of how to schedule data retrieval from a very large data set to produce a static chart. Example 3-6, “Employee Information Data for GlobalMega Corp.” on page 113describes a conceptual use case for employee information from a global company. Next, “Preparing to Visualize a Scheduled Data Chart for Very Large Data Sets” on page 113shows you how to prepare the template and “Step-by-step Walkthrough for a Scheduled Data Chart for Very Large Data Sets” on page 114takes you through the process of visualizing that data using the Visual Count widget.

To produce a non-interactive “infographic” chart that allows better performance with larger datasets, where the data is fetched periodically from the database reducing concurrent load, the SQL query of the LiveReport groups all the values.

Example 3-6: Employee Information Data for GlobalMega Corp.

Consider these first few rows and columns of a much larger dataset that lists several kinds of data for each employee in a huge global corporation. If we want a chart that shows the number of employees in each country, we can create a SQL query that groups by country and counts the rows for each. It is not important that the data is "live", in this case it is acceptable to see a snapshot of the data so long as it is updated on a daily basis.

Table 3-5: EmployeeData

EmployeeID	Country	Region	Gender	Function	City
12345	USA	Nebraska	Male	Engineering	Omaha
12346	USA	Texas	Male	Sales	Austin
12347	USA	New Jersey	Female	Finance	Newark
12348	France	Normandie	Female	Sales	Rouen
12349	France	Centre	Male	Engineering	Paris
12350	UK	Berkshire		Engineering	Reading
12351	UK	Hampshire	Female	Human Resources	Southampton
...	...	...	...	...	...

Preparing to Visualize a Scheduled Data Chart for Very Large Data Sets

Performance Considerations for a Scheduled Data Chart for Very Large Datasets

Note: This approach is most suitable for situations where you might expect a very large number of results.

- In this scenario, only a subset of the results are returned to the browser. The server-side SQL code in the LiveReport filters the results at the data layer.
- The data is fetched from the database periodically. You can define how frequently the scheduling report runs, achieving the best balance between placing heavy demands on the database versus how accurate and up-to-date the results are.
- It is a good practice to tune your LiveReport to only return the data that you need.

Step-by-step Walkthrough for a Scheduled Data Chart for Very Large Data Sets

Note: These examples have been tested for MSSQL and Oracle databases. If you are using a different type of database, the syntax may differ.

To step through the process of visualizing a very simple chart with static data

1. Create the LiveReport data source.

   For more information about how to create a LiveReport, see *OpenText Content Server - LiveReports (LLESREP-UGD)*.

   a. In Content Server, click **Add Item** to create a new LiveReport.
   b. Edit the report and provide a **Title** and a **Record Limit**. For this example, the Record Limit is 6.
   c. Edit the LiveReport to add the following SQL code to the SQL section:

   ```
   SELECT UPPER(FileType) AS Label, (SUM(DataSize)) AS Count
   FROM DVersData
   WHERE FileType IS NOT NULL
   AND FileType NOT LIKE '.%'
   AND FileType != ''
   GROUP BY UPPER(FileType)
   ORDER BY Count DESC
   ```

   d. In the **Report Format** area, select **Auto LiveReport** from the list.
   e. Click **Save and Exit**.

2. Create a WebReports object that generates a static data file and runs on a schedule:

   For more information about how to create a WebReport, see *OpenText Content Server - WebReports (LLESWEBR-UGD)*.

   a. In Content Server, click **Add Item** to create a new WebReports object.
   b. On the **Add WebReport** page, in the **Name** section, enter a name for the WebReport.
   c. In the **Data Source**section, click **Browse Content Server** and choose the LiveReport created in Step 1.
   d. In the **Reportview file** section, from the list, select `csv_scripted`.
   e. Click **Add**.
   f. Open the WebReport Properties, click the **Destination** tab, and make the following changes:

Field Name	Setting
Output Destination	Select the **Content Server** option.
Name	Type the name that you want for the CSV file. For example: `grouped output.csv`

Field Name	Setting
Duplicate Name Action	Select the **Add Version to Original** option.
Create In	Click **Browse** and select the location that you want.
Export MIME Type	From the list, click **text/csv**.
Export if no Data	Clear the check box.
Set Schedule	Select the check box.
Next run after	Select the date and time that you want the WebReport to first run. For example: `01/17/2017, 11:06 pm`
Run	Select how many times you want the WebReport to run. For example: 1
Every	Select how frequently you want the WebReport to run. For example: Select the **5-Minute Increments** option.

g. Click **Update**.

3. Create a WebReports object that will format the data in the required way to support the Visual Count Widget:

   a. In Content Server, click **Add Item** to create a new WebReports object.
   b. On the **Add WebReport** page, in the **Name** section, enter a name for the WebReport.
   c. In the **Data Source**section, click **Browse Content Server** and choose the CSV file created by the scheduled WebReport as a data source in Step 2.
   d. In the **Reportview file** section, from the list, select `widget_visual_count_grouping_in_data_source_simple`.
   e. Click **Add**.
   f. Edit the properties for this WebReport.
   g. Click the **Destination** tab and set the **Export MIME Type** to **application/json**.
   h. Click the **Source** tab and in the **Row Separate** box, click **Clear** and then click **LF**.
   i. Click **Update**.

4. Edit the reportview for this WebReport by making the following changes:

   a. In the Header Section, add the `COUNTFORMAT:"BYTES"` tag. The results should appear as follows:

```
[LL_WEBREPORT_INSERTJSON @FILTEREDCOUNT
ACTIVECOLUMN:"[LL_REPTAG_&active_column /]"
COUNTFORMAT:"BYTES"
GROUPBYINDATASOURCE:"TRUE" /]
```

for the following reasons:

- `ACTIVECOLUMN:"[LL_REPTAG_&active_column /]"` tells the system to use the active column specified when creating the perspective.
- `COUNTFORMAT:"BYTES"` tells the Filtered Count widget to format and display the data as `BYTES`. It will automatically append a unit. For example, KB, MB, or GB, to the values returned from the CSV.
- `GROUPBYINDATASOURCE:"TRUE"` tells the tag that it should expect pre-grouped data.

For more information about the `@FILTEREDCOUNT` directive, see the INSERTJSON tag.

b. Click **Add Version**.

5. Create a perspective to see the chart in Smart View:

a. Sign in to Perspective Manager and create a perspective.

For more information about how to create a perspective, see the *Online Help* available in Perspective Manager.

b. Configure a Visual Count widget with the following settings:

Name	Value
Widget Size	If using the Flow layout, choose either Single width or Double width.
Title	Storage by File Type
Chart Type	Donut
Source WebReport	<*YourWebReportID*> as created in Step 3.
Active Column	Label
Show Values as Percentages	Use the default value.
Group After	Use the default value.
Sort By	Use the default value.
Sort Direction	Use the default value.
Enable Tile Filtering Controls	Use the default value.
Enable Expanded View	Use the default value.

c. Sign in to the Smart View and verify that the Visual Count widget in the Perspective produces the donut chart.

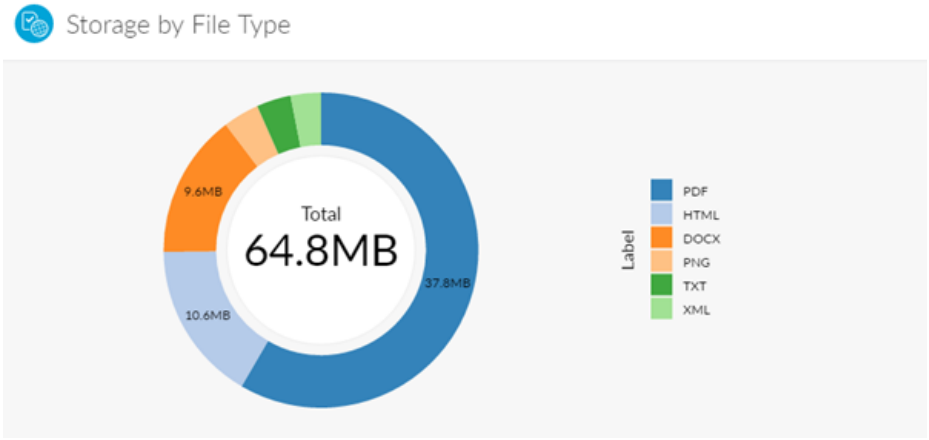

Figure 3-6: Scheduled Data Chart (Very Large Data Set)

3.4.4 Preparing WebReports for the Custom Buttons

In the Expanded view of the Visual Count widget, end users can click on the results in the chart to further filter the data. When you edit the widget, you have the option to configure one or more custom buttons to appear in the footer of the Expanded view. You can associate each custom button with a WebReport that can provide additional functionality using the filtered data. For example, bulk-changing document properties, exporting the data to a CSV file, or changing the status of a workflow; in other words, any operation that can be performed using WebReports tags.

To configure a custom button to appear in the Expanded view, you must prepare a WebReport for each button. For example, you could create the following WebReports:

- "Custom Button Example: Export to a CSV File" on page 119
- "Custom Button Example: Email Data to the User" on page 121
- "Custom Button Example: Save a Snapshot of the JSON Data" on page 123
- "Custom Button Example: Change the Owner of Filtered Documents" on page 127

Note: When using a custom button to retrieve data, the WebReport returns document metadata to the user, not grouped counts. This means that you must make sure to apply permissions filtering. Some data sources automatically apply permissions filtering or can be configured to do so. For more information about how to configure permissions filtering, see the documentation for your data source. If the data source does not have permission filtering, you can edit the **Row Section** of the reportview that generates the JSON data to exclude nodes to which the user does not have permissions by adding the following tag:

```
[LL_WEBREPORT_INCLUDEIF "[LL_REPTAG=DATAID
PERMCHECK:SEECONTENTS /]" == "TRUE" /]
```

Passing Parameters to a WebReport Associated with a Custom Button

When you configure a WebReport that will be associated with a custom button in the Expanded view of the Visual Count widget, the widget automatically passes request parameters that detail the current state of the application. For example, you can process the rows and have the custom button WebReport reflect the current Active Column selection and which filters are currently applied. The following table lists the valid parameters that are available and automatically passed:

Parameter Name	Parameter Tag and Description
Active Column	`[LL_REPTAG_&active_column /]` Name of the column in the data source that the WebReport uses for filtering the data.
Count Column	`[LL_REPTAG_&count_column /]` Name of the column in the data source that contains the count values.
Column Names	`[LL_REPTAG_&column_names /]` Comma-separated list that shows the current columns available for grouping and filtering.
View Value as Percentage	`[LL_REPTAG_&view_value_as_percentage /]` Boolean that indicates whether the values are displayed as a percentage of the total or as the actual count.
Group After	`[LL_REPTAG_&group_after /]` Current threshold that determines how many distinct data values should be displayed before grouping the remaining values under the label “Other”.
Sort Column	`[LL_REPTAG_&sort_by /]` Name of the column currently being used to sort the grouped data.
Sort Direction	`[LL_REPTAG_&sort_order /]` ASC or DESC string that indicates whether the grouped data is currently sorted in ascending order or descending order.
Total Count	`[LL_REPTAG_&total_count /]` Total count of the data rows, including applied filters.
Filters	`[LL_REPTAG_&fc_filters /]` JSON array of the currently applied filters.

Custom Button Example: Export to a CSV File

You can use the WebReports Export to Desktop feature to create a WebReport that will export the currently filtered rows in the Visual Count widget to a CSV file and then prompt the end user to download that CSV file. You can then associate this CSV WebReport to a custom button in the Expanded view of the Visual Count widget.

1. "To create a WebReport that generates the JSON data:" on page 119
2. "To create a WebReport that converts the JSON data to a CSV file:" on page 119
3. "To create a WebReport that calls the other two WebReports to generate and convert the data:" on page 120

To create a WebReport that generates the JSON data:

1. Create a WebReports node that uses the **widget_visual_count_grouping_on_client** default reportview.
2. Edit the WebReport properties as follows:
   a. Click the **Source** tab.
   b. In the **Data Source** section, select the data source to match the data source being used by the Visual Count widget from which you want to launch the custom button.
   c. Click **Update**.
3. Edit the WebReport reportview as follows:
   a. Add the following line to the `INSERTJSON` tag definition:

      ```
      FILTERBY:"[LL_REPTAG_&fc_filters /]"
      ```

      For example:

      ```
      [LL_WEBREPORT_INSERTJSON @FILTEREDCOUNT
      INCLUDECOLUMNS:"{'TYPE','FILESIZE'}"
      FILTERBY:[LL_REPTAG_&fc_filters /]
      GROUPBY:"FALSE"
      COUNTFORMAT:"SI" /]
      ```

   b. Add any additional configuration to match the WebReport that generates the data for the Visual Count widget. For example, set the `INCLUDECOLUMNS` or `EXCLUDECOLUMNS` values.

To create a WebReport that converts the JSON data to a CSV file:

1. Create a WebReports node that uses the **csv_scripted** default reportview.
2. Edit the WebReport properties:
   a. Click the **Destination** tab.
   b. In the **Export MIME Type** section, from the list, select **text/csv**.

c. Click **Update**.

To create a WebReport that calls the other two WebReports to generate and convert the data:

1. Create a WebReports node that uses the **blank_report** default reportview.
2. Edit the WebReport properties:
   a. Click the **Destination** tab.
   b. In the **Output Destination** section, from the list, select **Desktop**.
   c. In the **Download File Name** box, enter a name for the CSV file.
   d. In the **Export MIME Type** section, from the list, select **text/csv**.
   e. Click **Update**.
3. Edit the WebReport reportview:
   a. In the **Header Section**, add the following content:

```
[LL_REPTAG_$GETJSON RUNSWR:fc_filters:
[LL_REPTAG_&fc_filters /] TRIM FROMJSON SETVAR:chartData /]

[LL_REPTAG_$CREATECSV RUNSWR:DSREQUESTDATA:[LL_REPTAG_!
chartData ASSOC:data /]:DSFILETYPE:OSCRIPT SHOW /]
```

   Notes

   - This WebReport will call the `GETJSON` sub-WebReport to retrieve the data rows as JSON, passing in the current filter information. ThisWebReport has grouping disabled so that it will return every row of data rather than grouped counts. The WebReport will convert the data from JSON to an Oscript data format and set it as the `chartData` WebReports variable.
   - This WebReport will call the `CREATECSV` sub-WebReport and pass in the data rows using the `DSREQUESTDATA` feature. This sets the data source of the sub-WebReport using the data array from the chart data returned from the `GETJSON` report. The sub-WebReport will convert the JSON data and return the CSV data. The main WebReport will then return the output to the desktop as specified in the Destination settings and will prompt the user to download the file.

   b. Click **Add Version & Continue**.
4. Define the constants as follows:
   a. On the Header row, click the **Constant** link.
   b. In the new **Properties** window, on the **Constants** tab,
   c. In the top right corner of the tab, click **Extract Constants from Content** to automatically create two constants: `CREATECSV` and `GETJSON`.

d. For the `GETJSON` constant, in the **Constant Type** box, from the list, select **Content Server Object**, click **Browse**, and then select the WebReport created in "To create a WebReport that generates the JSON data:" on page 119.

e. For the `CREATECSV` constant, in the **Constant Type** box, from the list, select **Content Server Object**, click **Browse**, and then select the WebReport created in "To create a WebReport that converts the JSON data to a CSV file:" on page 119.

f. Click **Update** and close the window.

5. Return to the reportview editor and click **Add Version**.

6. Sign in to the Perspective Manager, for the Visual Count widget, configure the **Button WebReports** settings to associate this WebReport to the custom button.

7. Sign in to the Smart View and verify the custom button in the Expanded View of the Visual Count widget.

Custom Button Example: Email Data to the User

You can use the WebReports E-mail destination feature to create a WebReport that will email the currently filtered rows in the Visual Count widget to the current user. You can then associate this Email WebReport to a custom button in the Expanded view of the Visual Count widget.

1. "To create a WebReport that generates the JSON data:" on page 121
2. To create a WebReport that converts the JSON data to an HTML table: on page 122
3. "To create a WebReport that calls the other two WebReport to generate and convert the data:" on page 122

To create a WebReport that generates the JSON data:

1. Create a WebReports node that uses the **widget_visual_count_grouping_on_client** default reportview.

2. Edit the WebReport properties as follows:

   a. Click the **Source** tab.

   b. In the **Data Source** section, select the data source to match the data source being used by the Visual Count widget from which you want to launch the custom button.

   c. Click **Update**.

3. Edit the WebReport reportview as follows:

   a. Add the following line to the `INSERTJSON` tag definition:

   ```
   FILTERBY:"[LL_REPTAG_&fc_filters /]"
   ```

   For example:

```
[LL_WEBREPORT_INSERTJSON @FILTEREDCOUNT
INCLUDECOLUMNS:"{'TYPE','FILESIZE'}"
FILTERBY:[LL_REPTAG_&fc_filters /]
GROUPBY:"FALSE"
COUNTFORMAT:"SI" /]
```

b. Add any additional configuration to match the WebReport that generates the data for the Visual Count widget. For example, set the `INCLUDECOLUMNS` or `EXCLUDECOLUMNS` values.

To create a WebReport that converts the JSON data to an HTML table:

- Create a WebReports node using the **basic_scripted** default reportview.

To create a WebReport that calls the other two WebReport to generate and convert the data:

1. Create a WebReports node that uses the **blank_report** default reportview.
2. Edit the WebReport properties:
   a. Click the **Destination** tab.
   b. In the **Output Destination** section, select **E-mail**.
   c. In the **Email Address** box, enter the following:

   ```
   [LL_REPTAG_USERID USERINFO:MAILADDRESS /]
   ```

   which will resolve to the email address of the current user who runs this WebReport.
   d. In the **Email Subject** box, enter a subject.
   e. Click **Update**.
3. Edit the WebReport reportview:
   a. In the **Header Section**, add the following content:

   ```
   [LL_REPTAG_$GETJSON RUNSWR:fc_filters:
   [LL_REPTAG_&fc_filters /] TRIM FROMJSON SETVAR:chartData /]

   [LL_REPTAG_$CREATETABLE RUNSWR:DSREQUESTDATA:[LL_REPTAG_!
   chartData ASSOC:data /]:DSFILETYPE:OSCRIPT SHOW /]
   ```

   Notes

   - This WebReport will first call the `GETJSON` sub-WebReport to retrieve the data rows as JSON, passing in the current filter information. This WebReport has grouping disabled so that it will return every row of data rather than grouped counts. The WebReport will convert the data from JSON to an Oscript data format and set it as the `chartData` WebReports variable.
   - This WebReport will call the `CREATETABLE` sub-WebReport and pass in the data rows using the `DSREQUESTDATA` feature. This sets

the data source of the sub-WebReport to be the chart data returned from the `GETJSON` report. The sub-WebReport will convert the data to an HTML table and return it. The main WebReport will then email the output to the user as per the Destination settings.

b. Click **Add Version & Continue**.

4. Define the constants as follows:

   a. On the Header row, click the **Constant** link.

   b. In the new **Properties** window, on the **Constants** tab,

   c. In the top right corner of the tab, click **Extract Constants from Content**  to automatically create two constants: `CREATETABLE` and `GETJSON`.

   d. For the `GETJSON` constant, in the **Constant Type** box, from the list, select **Content Server Object**, click **Browse**, and then select the WebReport created in "To create a WebReport that generates the JSON data:" on page 121.

   e. For the `CREATETABLE` constant, in the **Constant Type** box, from the list, select **Content Server Object**, click **Browse**, and then select the WebReport created in To create a WebReport that converts the JSON data to an HTML table: on page 122..

   f. Click **Update** and close the window.

5. Return to the reportview editor and click **Add Version**.

6. Sign in to the Perspective Manager and open the perspective that includes the Visual Count widget to which you want to associate this custom button WebReport.

7. On the **Configure** tab, select the Visual Count widget and configure the **Button WebReports** settings to associate this WebReport to the custom button.

8. Sign in to the Smart View and verify the custom button in the Expanded View of the Visual Count widget.

Custom Button Example: Save a Snapshot of the JSON Data

You can use the WebReports Content Server destination feature to create a WebReport that will save a snapshot of the JSON data for later use. You can then associate this Snapshot WebReport to a custom button in the Expanded view of the Visual Count widget.

1. "To create a WebReport that generates the JSON data snapshot:" on page 124
2. "To create a WebReport that calls the JSON WebReport and redirects the user to the snapshot overview:" on page 124
3. "To create a WebReport that visualizes the snapshot as a chart for a new Visual Count widget :" on page 125

To create a WebReport that generates the JSON data snapshot:

1. Create a WebReports node that uses the **widget_visual_count_grouping_on_client** default reportview.
2. Edit the WebReport properties:
   a. Click the **Source** tab.
   b. In the **Data Source** section, select the data source to match the data source being used by the Visual Count widget from which you want to launch the custom button.
   c. Click the **Destination** tab.
   d. In the **Output Destination** section, from the list, select **Content Server**.
   e. In the **Name** box, enter a name for the JSON file.
   f. In the **Duplicate Name Action** section, select **Add Version to Original**.
   g. In the **Create In** section, click **Browse Content Server** and browse to the location that you want.
   h. Click **Update**.
3. Edit the WebReport reportview as follows:
   a. Add the following line to the `INSERTJSON` tag definition:

      ```
      FILTERBY:"[LL_REPTAG_&fc_filters /]"
      ```

      For example:

      ```
      [LL_WEBREPORT_INSERTJSON @FILTEREDCOUNT
      INCLUDECOLUMNS:"{'TYPE','FILESIZE'}"
      FILTERBY:[LL_REPTAG_&fc_filters /]
      GROUPBY:"FALSE"
      COUNTFORMAT:"SI" /]
      ```

   b. Add any additional configuration to match the WebReport that generates the data for the Visual Count widget. For example, set the `INCLUDECOLUMNS` or `EXCLUDECOLUMNS` values.

To create a WebReport that calls the JSON WebReport and redirects the user to the snapshot overview:

1. Create a WebReports node that uses the **blank_report** default reportview.
2. Edit the WebReport reportview:
   a. In the **Header Section**, add the following content:

      ```
      [LL_REPTAG_$CREATEJSONFILE RUNSWR:fc_filters:
      [LL_REPTAG_&fc_filters /]:SHOWNODEID SETVAR:nodeID /]
      [LL_REPTAG_URLPREFIX SETVAR:url /]
      [LL_REPTAG_'/app/nodes/' CONCATVAR:url /]
      [LL_REPTAG_!nodeID CONCATVAR:url /]
      [LL_REPTAG_!url REDIRECT /]
      ```

Notes

- This WebReport will first call the `CREATEJSONFILE` sub-WebReport to retrieve the data rows as JSON, passing in the current filter information. This WebReport has grouping disabled so that it will return every row of data rather than grouped counts. The WebReport will write the data to a new JSON file stored in Content Server . The DataID for the new file is returned using the `SHOWNODEID` option and set as a WebReports variable called `NodeID`.
- Next, we need to build a Smart View URL to this node and redirect the user to it using the `REDIRECT` sub-tag. Although this WebReport runs using a Classic View WebReports URL, this is hidden from the user who is only aware that they start and end in the Smart View. The resulting URL will take the user to the document overview screen for the new document and will resemble: *https://server/cgi/cs.exe/app/nodes/12345*

b. Click **Add Version & Continue**.

3. Edit the WebReport properties:

   a. Click the **Constants** tab.
   b. In the top right corner of the tab, click **Extract Constants** to automatically create the `CREATEJSONFILE` constant.
   c. For the `CREATEJSONFILE` constant, in the **Constant Type** box, change the value to **Content Server Object** and then browse to the WebReport created in "To create a WebReport that generates the JSON data snapshot:" on page 124.
   d. Click **Update** and close the window.

To create a WebReport that visualizes the snapshot as a chart for a new Visual Count widget :

1. Create a WebReports node that uses the **blank_report** default reportview.
2. Edit the WebReport properties:

   a. Click the **Source** tab.
   b. In the **Data Source** section, select the data source to match the data source being used by the Visual Count widget from which you want to launch the custom button.
   c. Click the **Destination** tab.
   d. In the **Export MIME Type** section, from the list, select **application/json**.
   e. Click **Update**.

3. Edit the WebReport reportview:

   a. In the **Header Section**, add the following content:

```
[LL_REPTAG_$GETJSONSNAPSHOT GETTEXT /]
```

Note: This WebReport uses the `GETTEXT` sub-tag to read in the JSON and then return it to the chart. The data in the JSON file is ungrouped, so you must choose an Active Column to group the data when the chart is run.

b. Click **Add Version & Continue**.

4. Define the constant as follows:

   a. On the Header row, click the **Constant** link.
   b. In the new **Properties** window, on the **Constants** tab,
   c. In the top right corner of the tab, click **Extract Constants from Content** to automatically create the `GETJSONSNAPSHOT` constant.
   d. For the `GETJSONSNAPSHOT` constant, in the **Constant Type** box, from the list, select **Content Server Object**, click **Browse**, and then select the WebReport created in "To create a WebReport that generates the JSON data snapshot:" on page 124.
   e. Click **Update** and close the window.

5. Return to the reportview editor and click **Add Version**.

6. Sign in to the Perspective Manager and open the perspective that includes the Visual Count widget to which you want to associate this Button WebReport.

7. On the **Configure** tab, select the Visual Count widget

   a. In the **Source WebReport** box, enter the ID for the WebReport created in this procedure.
   b. In the **Active Column** box, enter a column name.
   c. In the **Enable Tile Filtering Controls**, select **False**.
   d. In the **Enable Expanded View** box, select **False**.

      Note: Since this is a static snapshot of the data at a particular point in time, you must not enable the Tile filtering controls or the Expanded view.

   e. Click **Create**.

Custom Button Example: Change the Owner of Filtered Documents

You can use the WebReports `NODEACTION:OWNEDBY` tag to change the owner of all selected documents, if the current user has the appropriate permissions. You can then associate this Change Owner WebReport to a custom button in the Expanded view of the Visual Count widget.

1. "To create a WebReport that generates the JSON data:" on page 127
2. "To create a WebReport that calls the JSON WebReport and redirects the user to the snapshot overview:" on page 124

To create a WebReport that generates the JSON data:

1. Create a WebReports node that uses the **widget_visual_count_grouping_on_client** default reportview.
2. Edit the WebReport properties:
   a. Click the **Source** tab.
   b. In the **Data Source** section, select the data source to match the data source being used by the Visual Count widget from which you want to launch the custom button.
   c. Click **Update**.
3. Edit the WebReport reportview as follows:
   a. Add the following line to the `INSERTJSON` tag definition:

      ```
      FILTERBY:"[LL_REPTAG_&fc_filters /]"
      ```

      For example:

      ```
      [LL_WEBREPORT_INSERTJSON @FILTEREDCOUNT
      INCLUDECOLUMNS:"{'TYPE','FILESIZE'}"
      FILTERBY:[LL_REPTAG_&fc_filters /]
      GROUPBY:"FALSE"
      COUNTFORMAT:"SI" /]
      ```

   b. Add any additional configuration to match the WebReport that generates the data for the Visual Count widget. For example, set the `INCLUDECOLUMNS` or `EXCLUDECOLUMNS` values.

Note: For this example custom button, you must ensure that the data source consists of nodes and the output JSON data includes the DataID column.

To create a WebReport that calls the JSON WebReport to retrieve the data and change the owner:

1. Create a WebReports node that uses the **blank_report** default reportview.
2. Edit the WebReport reportview:

a. In the **Header Section**, add the following content:

```
[LL_REPTAG_$GETJSON RUNSWR:fc_filters:
[LL_REPTAG_&fc_filters /] TRIM FROMJSON SETVAR:chartData /]
[LL_REPTAG_!chartData ASSOC:data PLUCK:DataID
EACH:"NODEACTION:OWNEDBY" /]
```

Notes

- This WebReport will first call the `GETJSON` sub-WebReport to retrieve the data rows as JSON, passing in the current filter information. This WebReport has grouping disabled so that it will return every row of data rather than grouped counts. The WebReport will convert the JSON data to Oscript data format and set it as the `chartData` WebReports variable.
- This WebReport extracts a list of DataIDs in the JSON data and calls `NODEACTION:OWNEDBY` for each node in the list. If the current user has the appropriate permissions, they will be assigned as the new owner.

b. Click **Add Version & Continue**.

3. Define the constant:

a. On the Header row, click the **Constant** link.

b. In the new Properties window, on the **Constants** tab,

c. In the top right corner of the tab, click **Extract Constants from Content** to automatically create the `GETJSON` constant.

d. For the `GETJSON` constant, in the **Constant Type** box, from the list, select **Content Server Object**, click **Browse**, and then select the WebReport created in "To create a WebReport that generates the JSON data:" on page 127

e. Click **Update** and close the window.

4. Sign in to the Perspective Manager and open the perspective that includes the Visual Count widget to which you want to associate this Button WebReport.

5. On the **Configure** tab, select the Visual Count widget

a. In the **Source WebReport** box, enter the ID for the WebReport created in this procedure.

b. Click **Create**.

Chapter 4

Best Practices for WebReports Security Hardening

This section describes best practices for WebReports security hardening.

4.1 Preventing XSS-vulnerable Syntax in a Parameter Tag

WebReports reportviews and ActiveView templates are built using a tag-based syntax. These tags can be chained together to retrieve information or perform various actions in the system. The `Parameter` data tag, `[LL_REPTAG_&<parmName>/]`, provides access to a parameter value in the request. For example: the `[LL_REPTAG_&sortDirection /]` tag resolves to the value of the *`sortDirection`* parameter in the request. This syntax is usually used to surface data from the request, either to pass to additional WebReports (example: passing data to a SubWebReport) or to change aspects of the current ActiveView override (example: changing the current page size). The parameter values are resolved at runtime without any additional validation occurring behind the scenes.

If left unvalidated, the `Parameter` tag can create Cross-Site Scriping (XSS) vulnerabilities in a WebReports reportview or in an ActiveView template. For example: if you place a `Parameter` tag inside an HTML input field or in a Javascript block, but the parameter is unvalidated, the resulting syntax could be vulnerable to XSS. To illustrate, consider the following code:

```
<input type="hidden" name="count" value="[LL_REPTAG_&count /]">
```

In this case, the `[LL_REPTAG_&count /]` tag is replaced with the value of the *`count`* parameter from the request. This syntax is vulnerable to XSS because the count value is not validated. The *`count`* parameter value could contain HTML syntax to terminate the HTML input field early and inject additional HTML tags, including a `<script>` block.

As with any other web development, the users developing WebReports reportviews and ActiveView templates are responsible for validating their `Parameter` tags because they understand the context of how the parameter is used. Several sub-tags are available to help validate the passed values and are listed below:

- `CHECKURL` – checks a URL for potential XSS vulnerable syntax.
  Example: `[LL_REPTAG_&nextURL CHECKURL:"" /]`
- `ESCAPEFORJS` – converts a string using the same encoding as the specified Javascript escape function.
  Example: `[LL_REPTAG_&userName ESCAPEFORJS:HEX /]`
- `INT` – attempts to cast the data to an Integer. This is useful if the parameter value is expected to be an Integer.

Example: `[LL_REPTAG_&count INT /]`

- `TODATE` – attempts to cast the data to a Date.
  Example: `[LL_REPTAG_&startdate TODATE /]`
  Supports custom date formats as well.

Both the online Tag Guide and the *OpenText Content Server - WebReports (LLESWEBR-UGD)* provide a full listing of the available sub-tags as well as full details for each sub-tag.