

Overview

Smart List is a flexible component that allows you to build custom lists by using point and click configuration. It can handle Salesforce objects, files and custom data sources such as web services.

The component works with any Salesforce object and can be deployed on Lightning home, app, record and community pages as well as screen flows.

This allows you to get rid of most limitations of list views and related lists:

Capability	Smart List	OOB List
Highly Customizable Record Lists	Yes	Limited capability
Highly Customizable Tiles	Yes	No
Highly Customizable Search Component	Yes	Limited capability
Highly Customizable Files lists	Yes	No
Populate fields on file upload	Yes	No
Inline Edit		
Multiple Record Types	Yes	No
All Data Types	Yes	No
Custom Actions		
Complex logic on selected records	Yes	Limited capability
Complex logic on loaded records	Yes	No
Custom actions on all objects	Yes	No
Records Filtering		
Predefined filters	Yes	Limited capability
Visibility filters based on record ownership and role hierarchy	Yes	Limited capability
Records Search		
Customizable search form	Yes	No
SOSL Search on related lists	Yes	No
Easy search on Record Type, Owner, and Related record	Yes	No
Data Source		
Display data from external services or complex SOQL queries	Yes	No
Indirect Relationships such as display Contacts of the Account of a Case	Yes	Limited capability
Display records not shared with user	Yes	No
Available in Screen Flows	Yes	Limited capability
Trigger Programmatic list updates from a Lightning Web Component	Yes	No

Like list views and related lists, Smart Lists allow sorting, filtering and navigation to record detail. They also offer additional filtering capabilities with predefined and visibility filters based on record ownership and role hierarchy (My, My Team, My Subordinates and All)

The screenshot shows a 'Leads' list view with the following features highlighted:

- Sortable Columns:** A callout points to the column headers: Name, Rating, Owner, Email, and Record Type.
- Predefined Filters:** A callout points to a dropdown menu showing 'Converted Leads' and 'Leads in Progress' (which is selected).
- Visibility Filters:** A callout points to a dropdown menu showing 'All', 'My', 'My Team', and 'My Subordinates'.
- Hyperlinks with navigation to record detail:** A callout points to the names of the leads in the list.

At the top of the interface, there are buttons for 'Assign Selected Leads to Me', 'Create Tasks for Selected Leads', and 'New'. The list is currently sorted by 'Owner' and shows 20+ items.

Like Related Lists, Smart Lists can be displayed as Tiles with additional capabilities:

- Customizable Layout
- Customizable fields in Title
- Badge with Dynamic styling
- Sort
- Record selection

The screenshot shows a 'Sample Leads: Tiles 2x2' view with the following features highlighted:

- Customizable Title:** A callout points to the lead titles, such as 'Andy Young - Qualified'.
- Badge with Dynamic Styling:** A callout points to the status badges: 'Warm', 'Hot', and 'Cold'.
- Sortable Fields:** A callout points to a dropdown menu showing 'Email', 'Name', 'Owner', 'Phone', 'Rating', 'Record Type', and 'Status'.

At the top of the interface, there are buttons for 'Assign Selected Leads to Me', 'Create Tasks for Selected Leads', and 'New'. The list is currently sorted by 'Name' and shows 8+ items.

Unlike list views and related lists, the Search Form can be customized for displaying the fields you want to expose in the order relevant to your users.

It also allows you to easily search Owner, related records, and Record Type fields with lookups and picklists

The image shows a screenshot of a Salesforce search form titled "Filters" with a close button (X) in the top right corner. The form contains several search fields, each with a clear button (X) on its right side. On the left side of the form, there are four blue callout boxes with white text and arrows pointing to specific fields:

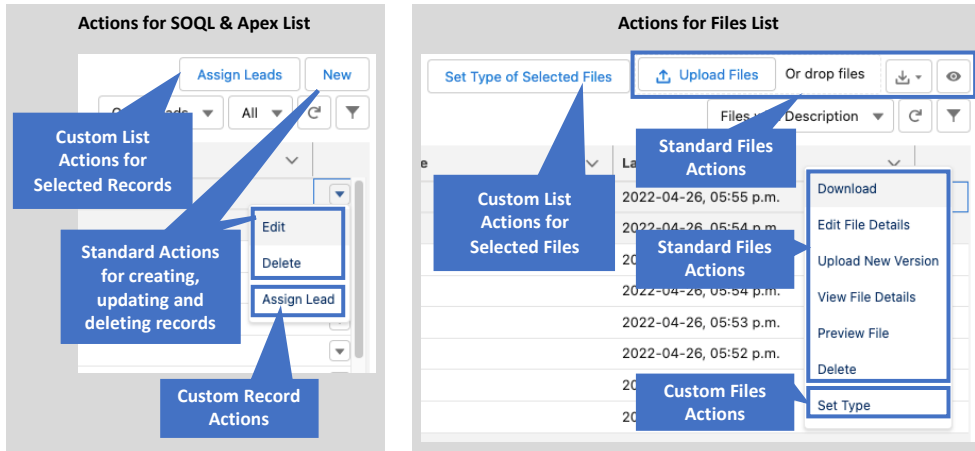
- SOSL Search on Record or File Content**: Points to the "Search Record" text input field.
- Search on Filterable Fields**: Points to the "Owner" dropdown menu.
- Lookup for searching Owner and related records**: Points to the "Search Users..." lookup field.
- Picklist for searching on record type**: Points to the "Record Type" section, which includes checkboxes for "Direct Sales" and "Partners".

The form fields include:

- Search Record**: A text input field.
- Owner**: A dropdown menu currently showing "Users".
- Search Users...**: A lookup field with a magnifying glass icon.
- Name**: A text input field.
- Record Type**: A section with two checkboxes: "Direct Sales" and "Partners".
- Email**: A text input field.
- Rating**: A section with three checkboxes: "Hot", "Warm", and "Cold".

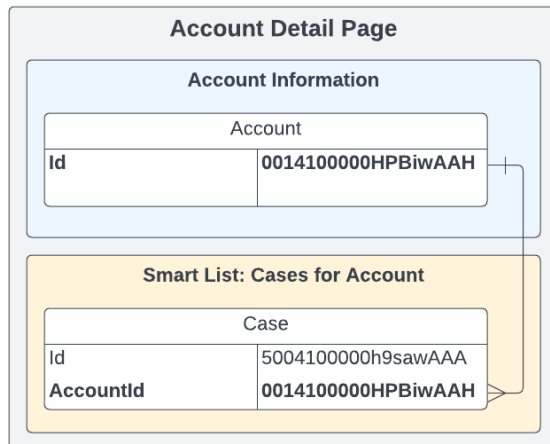
There are 3 types of Smart Lists:

- SOQL list for displaying and editing records from any Salesforce object
- Files list for managing files
- Apex Data Source for displaying data retrieved by an Apex class. This Apex class can be used for retrieving data with a Web Service or a complex SOQL query not handled by the standard SOQL list

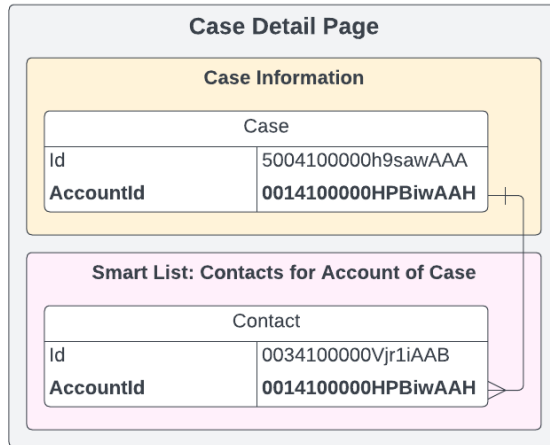


Smart Lists can be used for building child lists with 3 types of relationships:

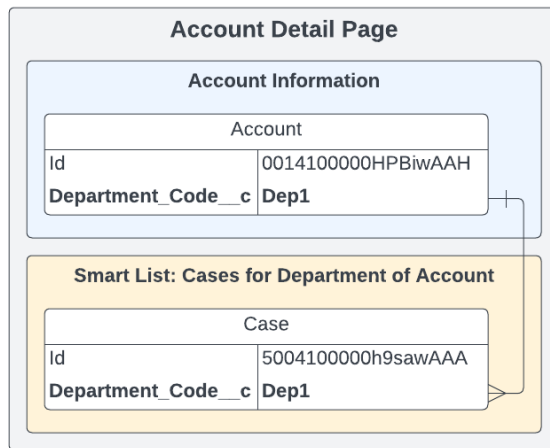
- Direct Relationship: records of the list are retrieved based on the Id of the parent record



- Indirect Relationship on Id: records of the list are retrieved based on id values stored present in both the parent record and the list records



- Indirect Relationship on Text Value: records of the list are retrieved based on text values present in both the parent record and the list records



Contents

How to use this document	8
Upgrade from Previous Versions.....	9
Upgrade from Summer 23	9
Upgrade from Spring 23	9
Upgrade from Summer 22	9
Quick Start.....	10
Example 1: List of Cases of an Account with inline edit, multiline cells, cell styling predefined filters, visibility filters, custom actions on selected records and custom New case action	11
Example 2: List of Files of an Account.....	25
Example 3: Tiles of Leads	28
Example 4: Leads search component with field value displayed as images and clickable label field navigating to a URL.....	33
Guided Setup.....	36
Step 1: Create a Smart List Definition.....	37
Step 2: Create Smart List Fields	43
Step 3 - Optional: Create Smart List Filters.....	48
Step 4 - Optional: Create Smart List Actions.....	48
Step 5 – Add the list to a target Page	50
Lightning App Builder Page.....	50
Digital Experience Page	51
Screenflow.....	53
Lookups.....	55
Search Customization	55
Wrap text mode.....	59
Sorting Customization.....	59
Files List Customization	60
Considerations for object, and field access	61
Considerations for standard record actions	61
External Object support.....	61
Features by List Type and Targets	63
Programmatic Updates of the Smart List with Lightning Messaging Service	65
Apex Data Source for Record Detail Page.....	65
Apex Data Source for Record Detail Page.....	65
Sample LMS Publisher	66

- Apex Data Sources..... 70
 - Apex Data Source for Record Detail Page..... 70
 - Apex Data Source for Home Page or Custom Tab 76
 - Pre-requisites for adding standard record actions to your list 81
 - Interface and Classes Reference 82
 - smartLists.SmartListApexSourceInterface2 Interface 82
 - smartLists.SmartListApexSourceGetPage Class..... 82
 - smartLists.SmartListController.FilterEntry Class 83
- Localization..... 84

How to use this document

If you are upgrading from Summer 22 or Spring 23, make sure to implement the steps described in the [Upgrade from Previous Versions](#) section

Visit this [Trailhead Group](#) if you want to collaborate and get updates on Smart Lists

Go to the [Quick Start](#) section if you want to learn Smart Lists configuration by building 4 samples lists

Go to the [Guided Setup](#) section if you need a step-by-step setup guide for all the use cases supported by Smart Lists

Go to the [Localization](#) section if you need to localize your Smart Lists for supporting multiple languages

Examples of Apex Data Sources can be found in the [Apex Data Sources](#) section

Upgrade from Previous Versions

Because of a Salesforce limitation, page layouts are not always updated during a package upgrade. If you don't see a field mentioned in this guide, edit the corresponding page layout and add it

Upgrade from Summer 23

This new field must be populated on your List Definition records:

- Filters Panel Layout must be set to 'Right - On Demand'

This new field must be populated on your List Action records:

- List Action Availability must be set to 'When Records are Selected'

Upgrade from Spring 23

These new fields must be populated on your List Definition records:

- Display Mode must be set to 'Table'
- Show SOSL Search must be set to 'In Filters Panel'

Upgrade from Summer 22

These new fields must be populated on your Field Definition records:

- Display in Filters Panel
- Display Position in Filters Panel
- Sortable in List

Apex Data Source classes must be updated with the new format see [Apex Data Source for Record Detail Page](#) and [Apex Data Source for Home Page or Custom Tab](#)

Quick Start

Learn Smart List configuration by configuring these 4 sample lists:

- [List of Cases of an Account with Inline Edit](#)
- [List of Files of an Account](#)
- [Tiles of Leads](#)
- [Search Screen of Leads](#)

Example 1: List of Cases of an Account with inline edit, multiline cells, cell styling predefined filters, visibility filters, custom actions on selected records and custom New case action

Close Selected Cases Reopen Selected Cases Create Account Task New						
3 items • Sorted by Last Modified Date <input type="text" value="Search this list..."/> Open Cases All Download Refresh Filter						
<input type="checkbox"/> Case Number	Subject	Description	Contact	Last Modified Date	Status	
2 <input type="checkbox"/> 00001000	Our samples have not been delive...	The customer didn't receive the samples: wrong shipping address	Jon Amos	02/19/2024, 01:34 PM	Working	<input type="checkbox"/>
3 <input type="checkbox"/> 00001002	Cannot track our order.	The customer is not able to track the order with the provided tracking number	Edward Stamos	02/17/2024, 02:07 PM	Escalated	<input type="checkbox"/>
4 <input type="checkbox"/> 00001001	The shirts we received are the wr...	They ordered XL and received XXL	Edward Stamos	02/17/2024, 01:59 PM	New	<input type="checkbox"/>

Step 1: Create a text formula field on the Case object for the dynamic styling

- Field Label: Status Style
- Field Name: Status_Style
- API Name: Status_Style__c
- Formula: IF (ISPICKVAL(Status, "Escalated"), "tc:firebrick;icn:utility:warning;icc:firebrick;icp:left", IF (ISPICKVAL(Status, "New"), "tc:#0033cc;icn:utility:alert;icc:#0033cc;icp:left", "tc:rgb(24,24,24)"))

Step 2: Create the list

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Definition
- Click New
- Use the following screenshot for creating the list definition:

▼ List Settings

Data Source Type	SOQL with Sharing	Display Mode	Table
List Label	Sample Cases List	List Icon	
Maximum Number of Records	2,000	Number of Records per Page	2
Disable Autoload on Initialization	<input type="checkbox"/>	Disable Paging	<input type="checkbox"/>
Selectable Rows	<input checked="" type="checkbox"/>	Max Row Selected	50
Filters Panel Layout	Right - On Demand	Filters Panel Max Height	150.00
Show SOSL Search	In Component	Default Sort Direction	desc
Export to CSV	<input checked="" type="checkbox"/>		

▼ Child Lists - Relationship Settings

Parent Id Field (Not for Files Lists)	AccountId	Indirect Relationship: Parent Key Field	
---------------------------------------	-----------	---	--

▼ Table Settings

Show Table Header	<input checked="" type="checkbox"/>	Wrap Column Headers	<input checked="" type="checkbox"/>
Show Row Number Column	<input type="checkbox"/>	Row Number Start	1
Table Height in Pixels	250.00	Wrap Text Max Lines	3

▶ Tiles Settings

▼ SOQL with/without Sharing & Apex Settings

SObject	Case	New Record Action Label	
Enable all Record Actions	<input checked="" type="checkbox"/>	Enable New Record Action	<input type="checkbox"/>
Enable Edit Record Action	<input type="checkbox"/>	Enable Delete Record Action	<input type="checkbox"/>
Default Visibility Filter	All	Visibility Filter - All	<input checked="" type="checkbox"/>
Visibility Filter - My	<input checked="" type="checkbox"/>	Visibility Filter - My Team	<input type="checkbox"/>
Visibility Filter - My Subordinates	<input checked="" type="checkbox"/>	Visibility Filter - My Queues	<input checked="" type="checkbox"/>

Notes:

- List Settings
 - Data Source Type: SOQL with Sharing for displaying the records visible by the user; SOQL without Sharing for ignoring the records visibility rules for the user
 - Display Mode: Table for displaying the list as a table
 - List Label is populated with 'Sample Case List'. If you leave this field empty, the list label will be the plural label of the list SObject, 'Cases' for this example
 - List Icon: if left empty, the icon of the base object is used. Can be used for adding a SLDS icon such as standard:lead or utility:cart
 - Filters Panels Layout: Specifies the position of the panels (left or right of the list) as well as if it is displayed all the time or when the Filters icon is clicked
 - Filters Panel Height: If a value (pixels) is specified, the panel will become scrollable if its heights exceeds this value
 - Show SOSL Search: In Filters Panel for displaying the search box in the Filters Panel, In Component for displaying it above the list, Not Displayed for removing the SOSL search
 - Export to CSV: the Export to CSV button is displayed in the list
- Child Lists - Relationship Settings
 - Parent Id Field: Field of the base object containing the id of the parent record for list of child records. Not needed if the list has no parent
- Table Settings
 - Wrap Column Header: Column headers are wrapped up to 3 lines if they don't fit in the column width; otherwise they are displayed on 1 line with ellipsis at the end
 - Table Height in Pixels: If specified, the height of the table is based on this value, otherwise, the height of the table is calculated based on Number of Records per Page; specifying a height is the preferred option if you have multiline cells
 - Wrap Text Max Lines: Cell content is wrapped up to the specified number of lines when Wrap is selected on the column. See [Wrap text mode](#)
- SOQL with/without Sharing & Apex Settings
 - SObject: Base object of the list
 - Enable All Record Actions: Allow to create, edit, and delete records in the list. Actions are displayed if the user has the corresponding access on the object. For example, if a user can create and edit records in the object but cannot delete them, the Delete action won't be available.
 - Visibility Filters fields: Check the filters you want to display in the list. Specify in Default Visibility Filter the filter that will be used when the list is displayed for the first time

Step 2: Create the fields

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Field
- Use the following table for creating the fields:

Field	List Settings			Filters Settings		
	Field Name	Display in List	Display Position in List	Sortable in List	Display in Filters	Display Position in Filters
CaseNumber	TRUE	0	Yes	Yes	0	Display Type Settings Display Type: Hyperlink to Detail Table Settings Column Width: 90
Subject	TRUE	1	Yes	Yes	3	Table Settings Inline Edit: TRUE
Description	TRUE	2	No	No		Table Settings Column Width: 250 Inline Edit: TRUE Wrap: TRUE
Contact.Name	TRUE	3	Yes	Yes	3	Field Lookup Subtitle Field: Title Display Type Settings Display Type: Hyperlink to Detail Filters Settings Lookup in Filters Panel: TRUE Table Settings Inline Edit: TRUE
LastModifiedDate	TRUE	4	Yes	Yes	4	List Settings Default Sort Field: TRUE Table Settings Field Alignment: Right
Status	TRUE	5	Yes	Yes	5	Fields Dynamic Style Field: Status_Style__c Table Settings Column Width: 160 Inline Edit: TRUE Field Alignment: Right
Owner	TRUE	6	Yes	Yes	6	Filters Settings Lookup in Filters Panel: TRUE Lookup Subtitle Field: Title
IsClosed	FALSE		No	No		

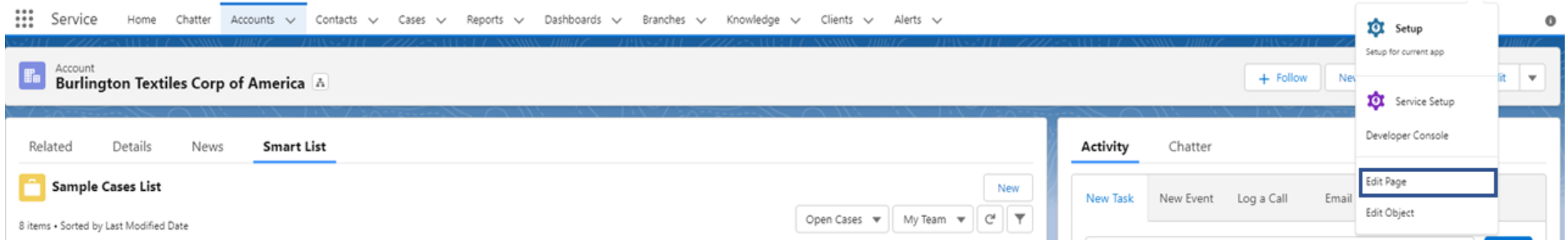
Notes:

- CaseNumber:
 - Rendered as a hyperlink to Case Detail: Display Type is Hyperlink to Detail
 - Initial column width is 90px; if no value is specified in Column Width, the width is determined by the component
- Subject: Editable in list because Inline Edit is checked
- Description:
 - Displayed in the list but not in the Filters Panel (Display in Filters = No)
 - Content will be wrapped on 3 lines if needed (Wrap = True; it is not sortable in the list (Sortable in List = No); the width of the column is 250 pixels (the width of the other columns is determined by the system because Column Width is left blank)
 - Editable in list because Inline Edit is checked
- Contact.Name:
 - Field Label is Contact because we don't want to use 'Full Name' which is the label defined at the object level
 - Display Type is Hyperlink to Detail because we want the field to be rendered as a hyperlink to the Contact detail page
 - Because Lookup in Filters Panel is checked and Lookup Subtitle Field which is set to Title, the field is searchable in the Filters Panel as a lookup where users can search Contacts by Name and Subtitle
 - Uncheck Lookup in Filters if you want to display a text box in the Filters Panel for searching the contacts by their name
 - Editable in list because Inline Edit is checked
- LastModifiedDate:
 - The content of the cell is aligned to the right because Field Alignment is set to right; Cells are aligned to the Left is no alignment is specified
- Status:
 - The content of the cell is styled based on the value returned by Status_Style__c:
 - tc: text color
 - icn: icon name see <https://www.lightningdesignsystem.com/icons/>
 - icc: icon color
 - icp: position of the icon relative to the text. The possible values are right, left and hidevalue if you want to display the icon without the value of the cell
 - Displayed in the list and in the Filters but not at the same position (Display Position in List = 5; Display Position in Filters = 1)
 - Editable in list because Inline Edit is checked

- Owner:
 - Because the field name is Owner, the list determines that the name of the owner must be displayed in the list and the field is searchable in Filters as a lookup where users can search the owner by name and by the field added in Lookup Subtitle Field which is the Title of Users in this example
 - If you want to search by Owner Name instead, set the field name to Owner.Name and uncheck Lookup in Filters Panel
- IsClosed: This field is not displayed in the list and the Filters Panel. It is added to the list because it is needed by a custom action of this example

Step 4: Test your page

- Make sure, the Apex Class 'smartLists.SmartListController' is accessible by your user
- Display an account record
- Setup / Edit Page



- In the Component widget on the left, select the Smart List component in Custom – Managed (1)
- Drag the Smart List component on the page (2)
- In List Definition Name (3), select the name of the list you created

The screenshot shows a CRM interface with a 'Components' sidebar on the left, a main account page for 'Burlington Textiles Corp of America', and a 'Smart Lists' section. A table of cases is displayed, and a configuration panel on the right shows the 'List Definition Name' set to 'SLAccountCases'.

Components Sidebar:

- Standard (0)
- Custom (0)
- Custom - Managed (2)
 - Smart Files List
 - Smart List** (1)

Account Page:

Account: Burlington Textiles Corp of America

Type: Customer - Direct | Phone: (336) 222-7000 | Website: www.burlington.com | Account Owner: Mark Si... | Account Site: Headquarter | Industry: Apparel

Smart Lists Section:

Related | Details | News | **Smart Lists**

Sample Cases List (2)

5 items • Sorted by Last Modified Date

Case Number	Subject	Description	Contact	Last Modified Date	Owner	Status
00001001	Performance inadequate f...		Avi Green	05/05/2023, 01:18 PM	Mark Simons	New
00001008	Customer service for port...		Lauren Boyle	05/05/2023, 01:18 PM	Mark Simons	New
00001020	Power generation below s...	Output is between 70-75% of expected level	Jack Rogers	04/05/2023, 12:51 PM	Electrical - Tier 1	New
00001052	Seeking guidance on elec...	Wiring instructions for secondary unit are not provided	Edna Rodgers	04/05/2023, 12:51 PM	Mark Simons	Working

All records are loaded

Configuration Panel (3):

Page > Smart List

List Definition Name: SLAccountCases

Displayed in tab

Set Component Visibility

Filters

+ Add Filter

- Activate the page if needed and save it
- Test the list. Change the parameters of the list definition to see how they affect the list

Step 5: Create predefined list filters

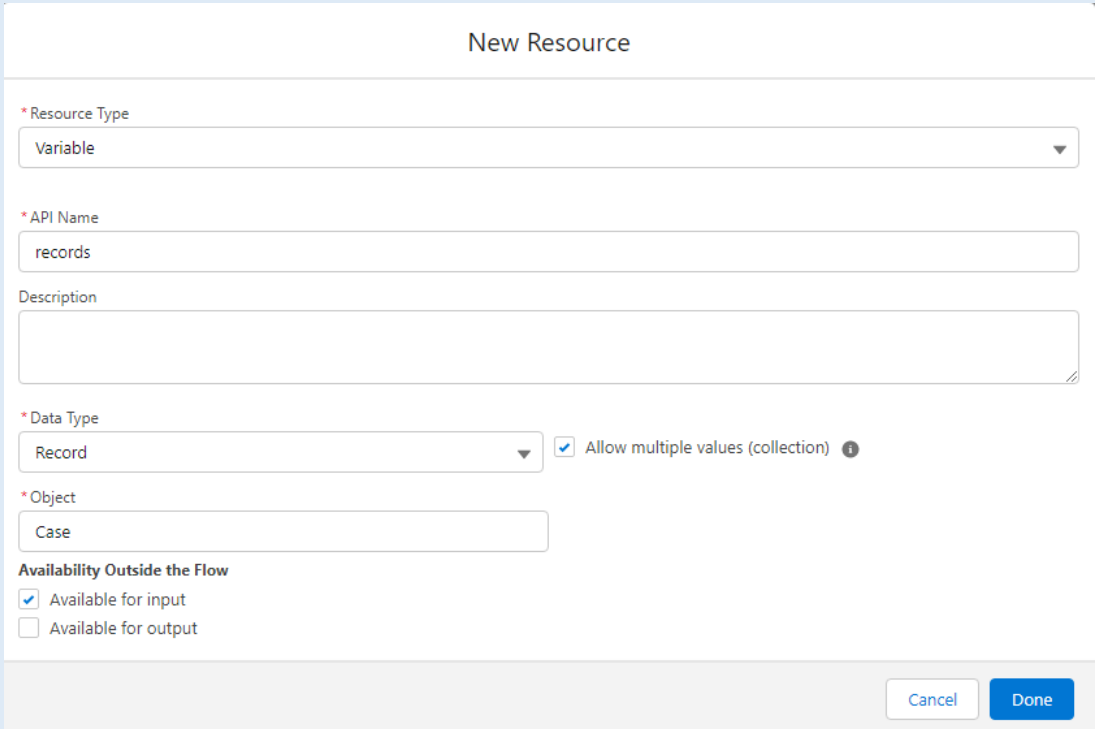
- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Filter
- Use the following table for creating the filters:

Filter Label	Default Filter	SOQL Filter
Cases I Created This Year	FALSE	CreatedDate = THIS_YEAR AND CreatedById = USERID
Closed Cases	FALSE	IsClosed = true
Open Cases	TRUE	IsClosed = false

- Notes:
 - Default Filter is used for specifying which filter is displayed when the list is displayed for the first time
 - SOQL Filter must be a valid SOQL WHERE clause
 - The Smart Lists variable USERID returns the Id of the running user
- Refresh the Account page to display the new filters

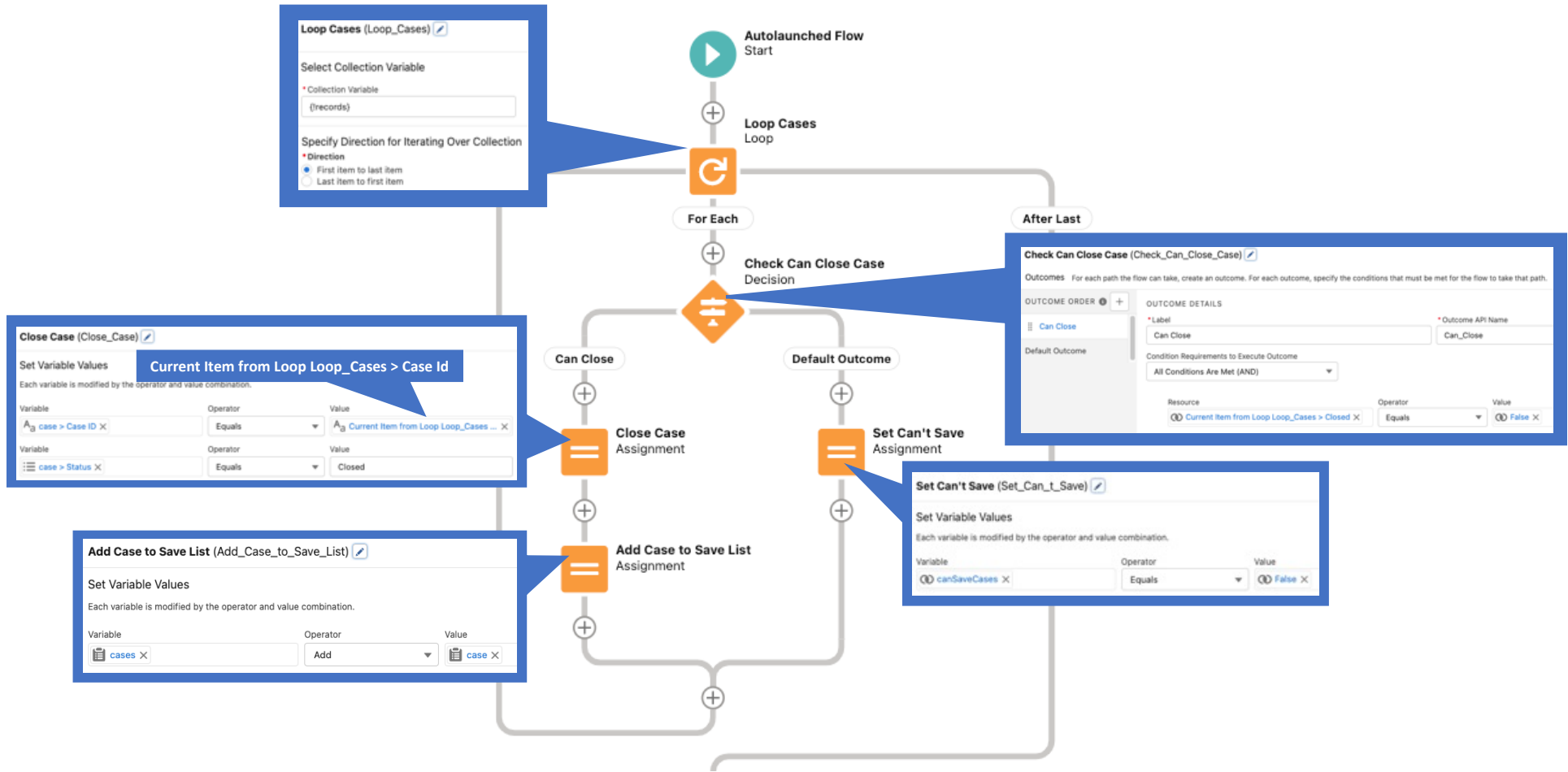
Step 6: Create custom row and list actions for closing the cases selected in the list

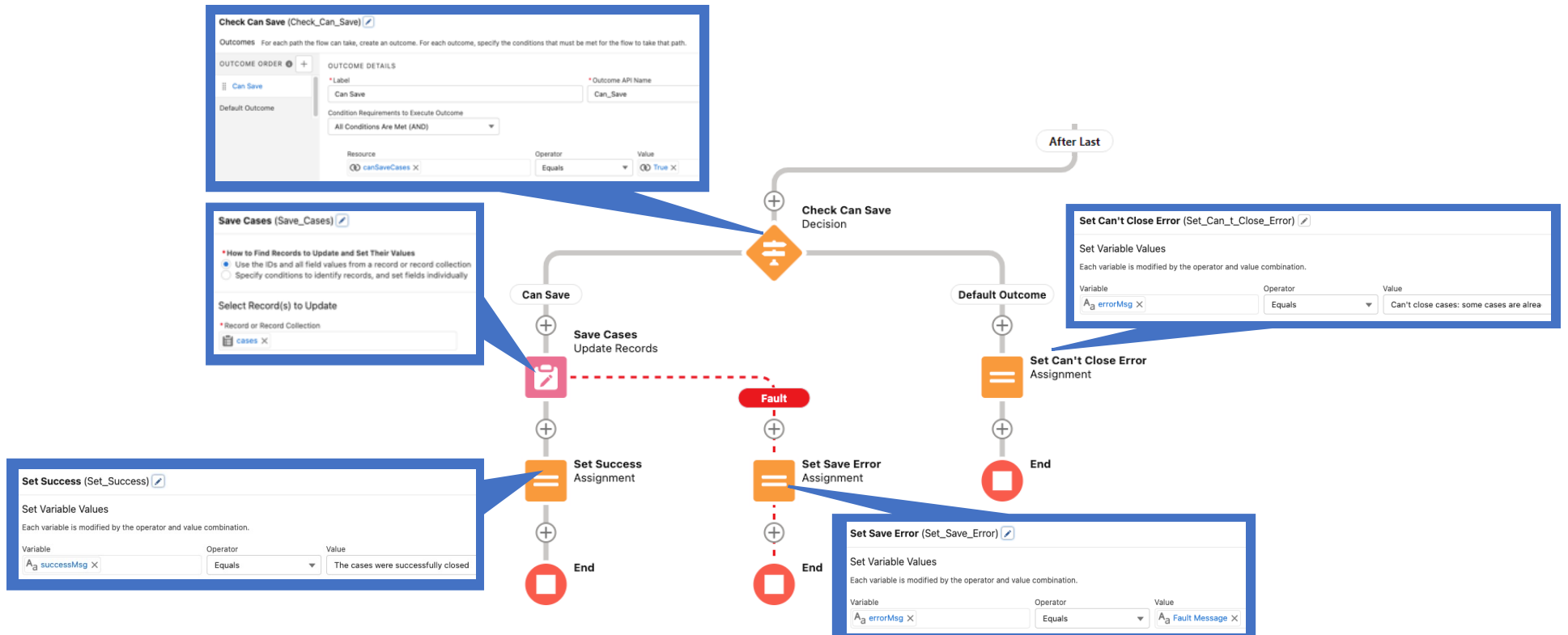
- Create a new auto-launched flow
- Create the following flow variables:

Variable	Type	Description
records	Collection of records – Available for input (Object = Case)	<p>Contains a list of selected records passed to the flow. Each record includes the Id field as well all the other fields defined in the list</p> 
parentId	Text – Available for input	Id of the parent account
successMsg (optional)	Text – Available for output	Message to display when the action is successful See Create Smart List Actions for more details

		<div style="border: 1px solid #ccc; padding: 10px;"> <h3 style="text-align: center; margin: 0;">New Resource</h3> <p>* Resource Type <input type="text" value="Variable"/></p> <p>* API Name <input type="text" value="successMsg"/></p> <p>Description <input style="width: 100%; height: 40px;" type="text"/></p> <p>* Data Type <input type="text" value="Text"/> <input type="checkbox"/> Allow multiple values (collection) ⓘ</p> <p>Default Value <input type="text" value="Enter value or search resources..."/></p> <p>Availability Outside the Flow</p> <p><input type="checkbox"/> Available for input</p> <p><input checked="" type="checkbox"/> Available for output</p> <p style="text-align: right;"> <input type="button" value="Cancel"/> <input type="button" value="Done"/> </p> </div>
errorMsg (optional)	Text – Available for output	Message to display when the action failed See Create Smart List Actions for more details
cases	Collection of Records (Object = Case)	Collection for storing the cases updated by the flow
case	Record (Object = Case)	Case to update in the database
canSaveCases	Boolean	Flag for tracking if the cases can be closed. Default value: <code>{!\$GlobalConstant.True}</code>

- Add the following elements to the flows:





- Save the flow as 'SLCases – Close Cases' and activate it
- Create a List Action: Setup / Custom Metadata Type / Click Manage Records in front of Smart List Action
- Use the following table for creating the list and row action:

Action Label	Type	Display Position	Refresh After Execution	Category	List Action Availability	Flow Name
Close Case	Row Action	0	Row	Autolaunched Flow		SLCases_Close_Cases
Close Selected Cases	List Action	0	List	Autolaunched Flow	When Records are Selected	SLCases_Close_Cases

Note: when List Action Availability is set to 'When records are Selected', the list action button can only be clicked when records are selected in the list

- Refresh the Account page to display the Actions
- Click the arrow on the right of a row to display the Row Actions and select the 'Close Case' menu item
- Select several records in the list by clicking the checkbox of the left column and click the 'Close Selected Cases' button
- Create a custom permission called 'Can Close Case' with API Name 'Can_Close_Case'
- Assign the custom permission to your profile or permission set

- Edit the List Actions and put 'Can_Close_Case' in the Custom Permission field
- Refresh the Account page and check that your actions are visible
- Remove the custom permission from your profile or permission set
- Refresh the Account page and check your actions are no longer visible

Step 7: Create a custom New action for creating new Cases

Smart Lists comes with a standard New record action that is enabled on the list definition. This action displays the record form of the page layout assigned to the running user. You can use a custom list action if you need to build your own new record form

- Create a new screenflow
- Create the following flow variables:

Variable	Type	Description
records	Collection of records – Available for input (Object = Case)	This parameter is optional for a New record action but may be needed for some use cases where you need to get the records loaded in the list
parentId	Text – Available for input	Id of the parent account
successMsg	Text – Available for output	Message to display when the action is successful
errorMsg	Text – Available for output	Message to display when the action failed


- Add the components needed for capturing the data and saving the case to the Screenflow
- Save and activate the screenflow
- Use the following table for creating the list action:

Action Label	Type	Display Position	Refresh After Execution	Category	List Action Availability	Flow Name
New	List Action	1	List	Screenflow	Always	API name of your screenflow



Notes:



- When List Action Availability is set to 'Always', the list action button can be clicked all the time
- Flow actions are displayed in modal dialogs. The UI of the dialogs can be customized
 - o Screenflow: Modal Height: specify the height of the modal in pixels. If left blank, the height of the modal is determined by the height of the flow
 - o Screenflow: Show Label in Modal Header: If checked, the label of the action is displayed in the header of the modal. Otherwise, no header is displayed. When you check this field, you may want to uncheck Show Header in the screens of your flow so that your label is the only title for the action.




Example 2: List of Files of an Account

 **Sample Files List**

3 items • Sorted by Last Modified Date

Update Description **Upload Files** Or drop files  

Search this list... Public Files All  

<input type="checkbox"/> Title	Description	File Privacy on Records	Last Modified Date	
<input type="checkbox"/> Smart Lists Release Notes 3.3	Smart Lists Release Notes 3.3	Visible to Anyone With Record Access	02/17/2024, 04:04 PM	
<input type="checkbox"/> Smart Lists Release Notes 3.2	Smart Lists Release Notes 3.2	Visible to Anyone With Record Access	02/17/2024, 04:03 PM	
<input type="checkbox"/> Smart Lists Release Notes 3.0	Smart Lists Release Notes 3.0	Visible to Anyone With Record Access	02/17/2024, 03:37 PM	

All records are loaded

Step 1: Create the list

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Definition
- Click New
- Use the following screenshot for creating the list definition:

▼ List Settings

Data Source Type	Files	Display Mode	Table
List Label	Sample Files List	List Icon	
Maximum Number of Records	2,000	Number of Records per Page	10
Disable Autoload on Initialization	<input type="checkbox"/>	Disable Paging	<input type="checkbox"/>
Selectable Rows	<input checked="" type="checkbox"/>	Max Row Selected	2,000
Filters Panel Layout	Left - On Demand	Filters Panel Max Height	
Show SOSL Search	In Component	Default Sort Direction	desc

► Child Lists - Relationship Settings

▼ Table Settings

Show Table Header	<input checked="" type="checkbox"/>	Wrap Column Headers	<input type="checkbox"/>
Show Row Number Column	<input type="checkbox"/>	Row Number Start	0
Table Height in Pixels		Wrap Text Max Lines	

► Tiles Settings

► SOQL with/without Sharing & Apex Settings

▼ Files Settings

Allowed Extensions

Notes:

- Allowed Extensions is used for specifying the file extensions allowed for upload:
 - If empty, no extension check is performed on upload
 - If not empty, the extension check is performed for all users without the custom permission Don't check file extension

Step 2: Create the fields

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Field
- Use the following table for creating the fields:

Field Name	List Settings			Filters Settings		Other
	Display in List	Display Position in List	Sortable in List	Display in Filters	Display Position in Filters	
Title	TRUE	0	Yes	Yes	0	Display Type Settings Display Type: TRUE Files Settings Editable in File Edit Form: Editable and required
Description	TRUE	1	Yes	Yes	1	Files Settings Editable in File Edit Form: Editable and required
SharingPrivacy	TRUE	2	Yes	Yes	2	Files Settings Editable in File Edit Form: Editable and required
LastModifiedDate	TRUE	3	Yes	Yes	3	List Settings Default Sort Field: TRUE

Notes:

- Display Type is not needed for Files list except for adding a link to the file preview popup. This popup is not available in Digital Experience sites
- The Editable in File Edit Form attribute is used for specifying which fields are included in the file edit form which is displayed when a file is uploaded, or its detail is updated

Step 3: Assign the following permissions to your profile or permission set

Label	API Name	Description
Don't check file extension	SmartFilesList_Don_t_check_file_extension	Bypass the file extension check for Upload File and Upload New Version actions
Download Files	SmartFilesList_Download_Files	Control access to File Download action
Edit File Details	SmartFilesList_Edit_File_Details	Control access to Edit File Details action
Preview Files	SmartFilesList_Preview_Files	Control access to Preview File action
Upload Files	SmartFilesList_Upload_Files	Control access to Upload File action
Upload New Version	SmartFilesList_Upload_New_Version	Control access to Upload New Version action
View File Details	SmartFilesList_View_File_Details	Control access to View File Details action


Step 4: Add a Smart Files List component to the Account Detail Page and select the name of the list you created in List Definition Name

Step 5: Test the component

Step 6: Add/Remove some of the above custom permissions and refresh the page to see how the list is affected

See [Files List Customization](#) for more details

Example 3: Tiles of Leads

 **Sample Leads: Tiles 2x2** Assign Selected Leads to Me Create Tasks for Selected Leads New

8+ items • Sorted by Name Leads in Progress My Subordinates ↕ ↑ ↻ ▼

<input type="checkbox"/> Andy Young • Qualified Owner: John Nguyen Email: a_young@dickenson.com Phone: (620) 241-6200 Record Type: Direct Sales Warm ▼	<input type="checkbox"/> Bertha Boxer • Qualified Owner: William Harper Email: bertha@fcf.net Phone: (555) 555-1212 Record Type: Partners Cold ▼
<input type="checkbox"/> Bertha Boxer • Contacted Owner: John Nguyen Email: bertha@fcf.net Phone: (850) 644-4200 Record Type: Direct Sales Hot ▼	<input type="checkbox"/> Bill Dadio Jr • Unqualified Owner: John Nguyen Email: bill_dadio@zenith.com Phone: (614) 431-5000 Record Type: Direct Sales Warm ▼
<input type="checkbox"/> Brenda McClure • Contacted Owner: John Nguyen Email: brenda@cardinal.net Phone: (847) 262-5000 Record Type: Direct Sales Cold ▼	<input type="checkbox"/> Carolyn Crenshaw • Unqualified Owner: William Harper Email: carolync@aceis.com Phone: (251) 679-2200 Record Type: Direct Sales Hot ▼
<input type="checkbox"/> David Monaco • Contacted Owner: William Harper Email: david@blues.com Phone: (033) 452-1299 Record Type: Direct Sales Warm ▼	<input type="checkbox"/> Jack Rogers • Qualified Owner: William Harper Email: jrogers@btca.com Phone: (336) 222-7000 Record Type: Direct Sales Warm ▼

Load More Load All

Step 1: Create the list

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Definition
- Click New
- Use the following screenshot for creating the list definition:

▼ List Settings

Data Source Type	SOQL without Sharing	Display Mode	Tiles
List Label	Sample Leads: Tiles 2x2	List Icon	
Maximum Number of Records	35	Number of Records per Page	8
Disable Autoload on Initialization	<input type="checkbox"/>	Disable Paging	<input type="checkbox"/>
Selectable Rows	<input checked="" type="checkbox"/>	Max Row Selected	10
Filters Panel Layout	Left - On Demand	Filters Panel Max Height	249.50
Show SOSL Search	Not Displayed	Default Sort Direction	asc

▶ Child Lists - Relationship Settings

▶ Table Settings

▼ Tiles Settings

Tile Layout 2x2

▼ SOQL with/without Sharing & Apex Settings

SObject	Lead		
Enable all Record Actions	<input checked="" type="checkbox"/>	Enable New Record Action	<input type="checkbox"/>
Enable Edit Record Action	<input type="checkbox"/>	Enable Delete Record Action	<input type="checkbox"/>
Default Visibility Filter	My Subordinates	Visibility Filter - All	<input checked="" type="checkbox"/>
Visibility Filter - My	<input checked="" type="checkbox"/>	Visibility Filter - My Team	<input checked="" type="checkbox"/>
Visibility Filter - My Subordinates	<input checked="" type="checkbox"/>	Visibility Filter - My Queues	<input type="checkbox"/>

Notes:

- Data Source Type set to 'SOQL without Sharing': running users can see records that are not shared with them. This setting is only meant to be used for very specific use cases. SOQL with Sharing should be the preferred option
- Tile Layout set to 2x2: display 2 tiles per row and 2 fields per tile
- Filters Panels Max Height set to 249.50: the height of the Filters Panel is limited to 249.5px; users must scroll to see some of the filters
- Show SOSL Search is set to 'Not Displayed': the SOSL search box is never displayed

Step 2: Create a formula field on the Lead object for styling the badge

Field Label: Rating Badge Style

Field Name: Rating_Badge_Style__c

Data Type: Formula Text

Formula:

```
IF( ISPICKVAL(Rating, "Hot"), "bc:#2e844a;tc:white", IF(ISPICKVAL(Rating, "Warm"), "bc:#feca39;tc:black", "bc:#2172d5;tc:white"))
```

bc: background color of the badge; must be a valid CSS color such as white or RGB(128,128,128) or hex color

tc: color of the text if the badge; must be a valid CSS color such as white or RGB(128,128,128) or hex color

if an invalid style is returned by the formula field, the badge will be displayed with a blue background and a white text

Step 3: Create the fields

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Field
- Use the following table for creating the fields:

Field Name	List Settings			Filters Settings		Other
	Display in List	Display Position in List	Sortable in List	Display in Filters	Display Position in Filters	
Name	TRUE	0	Yes	Yes	0	Display Type Settings Display Type: Hyperlink to Detail List Settings Default Sort Field: TRUE Tile Settings Display as Tile Header: TRUE
Status	TRUE	1	Yes	Yes	1	Tile Settings Display as Tile Header: TRUE
Rating	TRUE	2	Yes	Yes	2	Tile Settings Display as Tile Header: TRUE Display as Badge: TRUE Badge Style Field: Rating_Badge_Style__c
Owner	TRUE	3	Yes	Yes	3	Filters Settings Lookup in Filters Panel: TRUE Lookup Subtitle Field: Title
Phone	TRUE	4	Yes	Yes	4	
Email	TRUE	5	Yes	Yes	5	
RecordType	TRUE	6	Yes	Yes	6	

Notes:

- Name: This field is displayed in the Tile Header because Display in Tile Header is checked
- Status: This field is displayed after Name in the Tile Header because Display in Tile Header is checked and its Display Position in List is greater
- Rating: This field is displayed in the Tile Header as a dynamically style badge based on the style returned by the formula field Rating_Badge_Style__c
- RecordType:
 - Because the field name is RecordType, the list determines that the name of the record type must be displayed in the list and the field is searchable in Filters as a picklist containing all the active record type values

Step 4: Add a Smart List component to the Home Page and select the name of the list you created in List Definition Name

Step 5: Change the value of Tile Layout on the list definition and refresh the page to see how the layout of the Tiles is affected

Example 4: Leads search component with field value displayed as images and clickable label field navigating to a URL

Search Leads

8 items • Sorted by Name • Filtered by Owner

Add to Campaign

Search this list... All ↕ ↻

Filters

Name

Rating

- Hot
- Warm
- Cold

Owner

Queues

North East

Company

Cancel Clear All Filters Apply

<input type="checkbox"/>	Name ↑	Rating	HTML Rating	Owner	Email	Company
<input type="checkbox"/>	Andy Smith			North East	asmith@ubuilding.com	Universal Technologies
<input type="checkbox"/>	Jim Steele			North East	jimsteele@biglife.com	BigLife Inc.
<input type="checkbox"/>	John Gardner			North East	john@3cs.com	3C Systems
<input type="checkbox"/>	Mark Li	?		North East	mark@3cs.com	3C Systems
<input type="checkbox"/>	Mike Braund			North East	likeb@metro.com	Metropolitan Health Ser...
<input type="checkbox"/>	Norm May			North East	norm_may@greenwich....	3C Systems
<input type="checkbox"/>	Patricia Feager			North East	patricia_feager@is.com	3C Systems
<input type="checkbox"/>	Violet Maccleod			North East	violetm@emersontransp...	Emerson Transport

All records are loaded

Step 1: Create the list

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Definition
- Click New
- Use the following screenshot for creating the list definition:

▼ List Settings

Data Source Type	SOQL without Sharing	Display Mode	Table
List Label	Search Leads	List Icon	
Maximum Number of Records	2,000	Number of Records per Page	10
Disable Autoload on Initialization	<input checked="" type="checkbox"/>	Disable Paging	<input checked="" type="checkbox"/>
Selectable Rows	<input checked="" type="checkbox"/>	Max Row Selected	50
Filters Panel Layout	Left - All the Time	Filters Panel Max Height	350.00
Show SOSL Search	In Component	Default Sort Direction	asc

▶ Child Lists - Relationship Settings

▶ Table Settings

▼ Tiles Settings

Tile Layout

▼ SOQL with/without Sharing & Apex Settings

SObject	Lead		
Enable all Record Actions	<input type="checkbox"/>	Enable New Record Action	<input type="checkbox"/>
Enable Edit Record Action	<input type="checkbox"/>	Enable Delete Record Action	<input type="checkbox"/>
Default Visibility Filter	All	Visibility Filter - All	<input checked="" type="checkbox"/>
Visibility Filter - My	<input checked="" type="checkbox"/>	Visibility Filter - My Team	<input type="checkbox"/>
Visibility Filter - My Subordinates	<input checked="" type="checkbox"/>	Visibility Filter - My Queues	<input checked="" type="checkbox"/>

Notes:

- Because Disable Autoload on Initialization is checked, no records are loaded when the list is displayed for the first time
- Because Disable Paging is checked, all records matching the search criteria are loaded at once
- Because Filters Panel Layout is set to 'Left – All the Time' the Filters Panel is displayed on the left of the list and cannot be closed

Step 2: Create 2 formula fields on the Lead object

Dynamic styling of a cell

Field Label: Rating Style

Field Name: Rating_Style__c

Data Type: Formula Text

Formula:

```
IF ( ISPICKVAL( Rating , 'Hot'), "icn:utility:priority;icc:seagreen;icp:hidevalue", (IF ( ISPICKVAL( Rating , 'Warm'),  
"icn:utility:priority;icc:rgb(255, 215, 0);icp:hidevalue", (IF ( ISPICKVAL( Rating , 'Cold'), "icn:utility:priority;icc:blue;icp:hidevalue",  
"icn:utility:question_mark;icp:hidevalue"))))))
```

- icn: icon name
- icc: icon color
- icp: icon position; set to hidevalue for displaying the icon without the value; can also be left or right

Dynamic HTML of a cell

Field Label: HTML Rating

Field Name: HTML_Rating__c

Data Type: Formula Text

Formula:

```
CASE(Rating,  
"Hot", IMAGE("/resource/GraphicsPackNew/silk/16/silk/flag_green.png", "Hot"),  
"Warm", IMAGE("/resource/GraphicsPackNew/silk/16/silk/flag_yellow.png", "Warm"),  
"Cold", IMAGE("/resource/GraphicsPackNew/silk/16/silk/flag_blue.png", "Cold"), "")
```

Note: the images of the formula are not part of the package. You can either install the Graphics Pack from the AppExchange or add your own images to a static resource and update the formula accordingly. The height of the images must be 19px or lower

Step 3: Create the fields

- Setup / Custom Metadata Type / Click Manage Records in front of Smart List Field
- Use the following table for creating the fields:

Field Name	List Settings			Filters Settings		Other
	Display in List	Display Position in List	Sortable in List	Display in Filters	Display Position in Filters	
Name	TRUE	0	Yes	Yes	0	Display Type Settings Display Type: Hyperlink to Detail List Settings Default Sort Field: TRUE
Rating	TRUE	1	Yes	Yes	1	Field Dynamic Style Field: Rating_Style__c Table Settings Field Alignment: Center
HTML_Rating__c	TRUE	2	Yes	No		
Owner	TRUE	3	Yes	Yes	3	Filters Settings Lookup in Filters Panel: TRUE Lookup Subtitle Field: Title
Email	TRUE	4	Yes	Yes	4	
Company	TRUE	5	Yes	Yes	5	Display Type Settings Display Type: URL with Label URL with Label Value: Website

Notes:

- Rating:
 - The field is styled based on the formula field form Dynamic Style Field
 - The value of the field is centered in the cells with Field Alignment = Center
- Company: because Display Type is set to 'URL with Label', users are navigated to the company web site if they click on the company name
 - Make sure that Website contains a value otherwise, the field will not be clickable

Step 4: Add a Smart List component to the Home Page and select the name of the list you created in List Definition Name

Guided Setup

Step 1: Create a Smart List Definition

Smart List Definition Detail

- Enter unique values in Label & Smart List Definition Name

List Settings

This section contains the settings for all list types

- Data Source Type:
 - SOQL with Sharing: display records of an object and enforce the records visibility rules of the running user
 - SOQL without Sharing: display records of an object and bypass the records visibility rules of the running user
 - Files: display files related to a parent record
 - Apex Data Source: custom Apex Data Provider
- Display Mode: Table or Tile
- List Label
 - Leave empty if you want to use the default value (Files lists or lists with SObject)
 - Default value for list with a SObject: plural label of the SObject. Example: Cases if SObject is set to Case
 - Default value for Files list: Files
 - You can override the default label by entering a value or \$Label.CustomLabelName if you want to use a translatable label. Example: \$Label.CasesLists for using a custom label named CaseLists

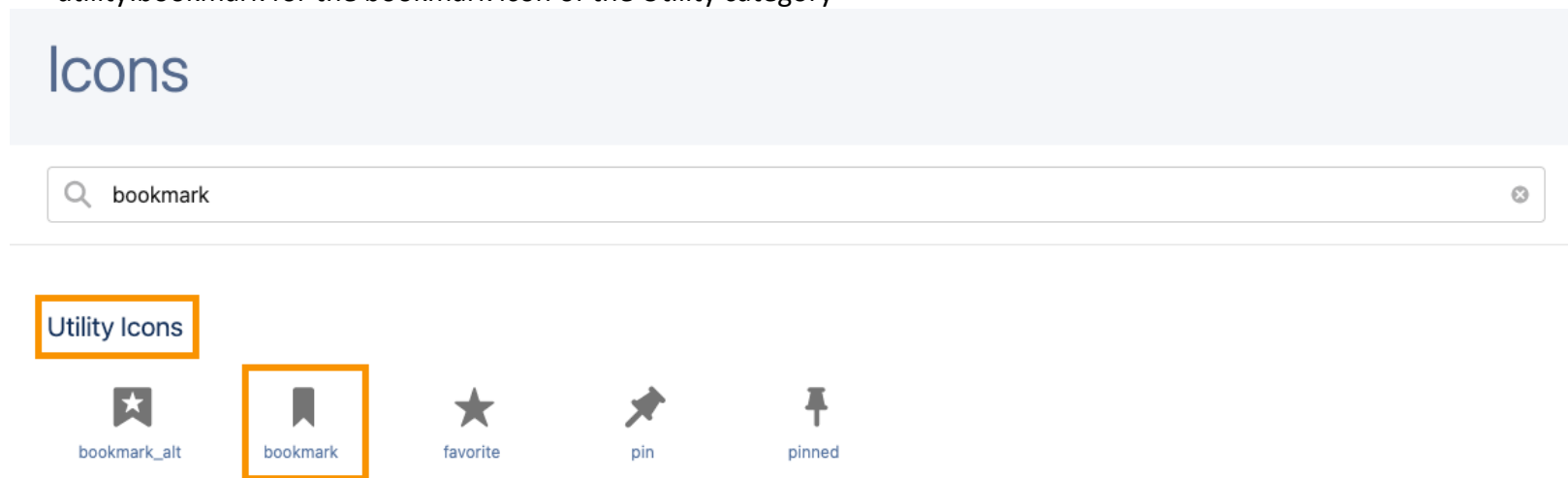
Custom Label Detail

[Edit](#) [Delete](#)

Short Description	CasesLists	Name	CasesLists
Language	English		
Categories			
Value	Cases Smart List		

- You need to provide a value for Apex lists not associated to a SObject
- List Icon
 - Leave empty if you want to use the default value (Files lists or lists with SObject)
 - Default value for list with a SObject associated to a tab: icon of the object
 - Default value for list with a SObject not associated to a tab (Campaign Member, Opportunity Product...): Salesforce default icon. See below for setting your own icon
 - Default value for Files list: standard Files icon

- You can override the default icon by entering the name of a Salesforce icon: <https://www.lightningdesignsystem.com/icons/>
 - The icon name must respect the following syntax: name of the category in lower case + : + name of the icon. Example, utility:bookmark for the bookmark icon of the Utility category



- You need to provide a value for Apex lists not associated to a SObject
 - No icon is displayed is you provide an invalid value
- Records management
 - Maximum Number of Records: maximum number records that can be displayed in a list
 - Number of Records per Page: number of records retrieved when the user scrolls in the list; for tables, used for determining the height of the list
 - Disable Autoload on Initialization: if checked, no records are loaded when the list is displayed for the first time
 - Disable Paging: if checked, when the list is displayed for the first time or a search is made in the Filters Panels, the list will try to load all the records matching the criteria. If the number of returned records is higher than the value defined in Maximum Number of Records, the number of loaded records will be limited this value. Disabling paging can impact the performance of the list and must be tested with the target datasets
- Row Selection
 - Selectable Rows: checked if users can select rows in the list; required if you want to use list level action
 - Max Row Selected: value greater than 0; radio buttons are displayed if 1 is entered, otherwise checkboxes are displayed
- Filters Panel Layout, Filters Panel Max Height & Show SOSL Search: see [Search Customization](#)
- Default Sort Direction: see [Sorting Customization](#)
- Export to CSV: the Export to CSV button is displayed and can be used for exporting the records to a CSV file

Child Lists - Relationship Settings

This section is used for configuring child lists (list of records related to a parent record). Examples: Cases of an Account or Files of an Account

2 types of relationships are available:

- Direct relationship: the id of the parent record is used for retrieving the child records displayed in the list:
 - Parent Id Field: API name of the field containing the Id of the parent record on the list object. For example, AccountId for a list of child Cases of a parent Account because Case.AccountId contains the value of the parent record. Only supported for SOQL & Apex lists
- Indirect relationship on Id: an Id field of the parent record is used for retrieving the records of the list
 - For SOQL and Apex list, the relationship can be built on any datatype (text, number...); for Files lists, the relationship can only use Id fields
 - Parent Id Field: API name of the field containing the Id of the parent record on the list object. Only supported for SOQL & Apex lists
 - Indirect Relationship: Parent Key Field: API name of the field of the parent object containing the id of the list records
- Notes:
 - Parent Id Field must be a field of the base object of the list
 - Formula fields can be used for retrieving related ids for Parent Id Field and Parent Key Field. For Id based relationship, the formula must use CASESAFEID to return the 18 characters ids expected by Smart Lists

Examples:

- List of Cases for an Account (list on an Account record page):
 - Parent Id Field: AccountId (name of the Case field containing the Id of the parent Account)
- List of Files for an Account (deployed on an Account record page):
 - Parent Id Field: not used
- List of Contacts related to the Account of a parent Case (list on a Case record page)
 - Indirect Relationship: Parent Key Field: AccountId because Case.AccountId contains the Id of the Account related to the Case
 - Parent Id Field: AccountId because Contact.AccountId contains the Id of the Account related to the Contact
- List of Contacts related to the department of a parent Case (deployed on a Case record page)
 - A custom text field Department__c has been created on Case & Contact for storing the code of the department
 - Indirect Relationship: Parent Key Field: Department__c because Case. Department__c contains the code of the department of the Case
 - Parent Id Field: Department__c because Contact. Department__c contains the code of the department of the Contact
- List of Files related to the Account of a parent Case (list on a Case record page)
 - Indirect Relationship: Parent Key Field: AccountId because Case.AccountId contains the Id of the Account related to the Case
 - Parent Id Field: not used

Table Settings

This section is used when Display Mode is set to 'Table' and contains the parameters of the table:

- Show Table Header: display the names of the columns in a table header
- Wrap Column Headers: the headers are wrapped up to 3 lines if they don't fit in the width of the column
- Show Row Number Column: if checked, a column is added on the left of the table and contains the number of the row
- Table Height in Pixels: height of the table in pixels; if not specified the height of the table is calculated based on Number of Records per Page
- Row Number Start: if Show Row Number Column is checked, specifies the number of the first row
- Wrap Text Max Lines: see [Wrap textmode](#)

Tiles Settings

This section is used when Display Mode is set to 'Tiles' and contains the parameters of the tiles:

- Tile Layout: controls the layout of the tile. 3x1 means that each row of the component will contains 3 tiles and 1 field will be displayed in a row of a tile
- Example: 2 tiles per row of the component

Sample Leads: Tiles 2x2
8+ Items • Sorted by Name

Assign Selected Leads to Me Create Tasks for Selected Leads New

Leads in Progress My Subordinates ↕ ↕ ↕ ↕ ↕ ↕

<input type="checkbox"/> Andy Young - Qualified Owner: John Nguyen Email: a_young@dickenson.com Phone: (620) 241-6200 Record Type: Direct Sales Warm	<input type="checkbox"/> Bertha Boxer - Qualified Owner: William Harper Email: bertha@cof.net Phone: (555) 555-1212 Record Type: Partners Cold
--	--

- Example: 2 fields per row of a tile

Sample Leads: Tiles 2x2
8+ Items • Sorted by Name

Assign Selected Leads to Me Create Tasks for Selected Leads New

Leads in Progress My Subordinates ↕ ↕ ↕ ↕ ↕ ↕

<input type="checkbox"/> Andy Young - Qualified Owner: John Nguyen Email: a_young@dickenson.com Phone: (620) 241-6200 Record Type: Direct Sales Warm	<input type="checkbox"/> Bertha Boxer - Qualified Owner: William Harper Email: bertha@cof.net Phone: (555) 555-1212 Record Type: Partners Cold
--	--

SOQL with/without Sharing & Apex Settings

- SObjectName: name of the SObject of the list. For Apex lists, only needed if you want to add record actions to your list
- Record Actions:
 - Control the standard record actions: New, Edit, and Delete Record. These actions are not displayed if the running user does not have the corresponding access on the object
 - Enable all Record Actions: check if you want to allow users to create, update and delete records in the list
 - Enable New, Edit, Delete Record Action: check if you want to specify which record actions are available in the list
 - New Record Action Label: use your own label for the New record action
- Visibility Filters:
 - Apart from My Queues, all Visibility Filters are filtering records based on record ownership and role hierarchy:

User: CEO
Role: CEO

User: VP Service
Role: VP

User: Director Service
Role: Director

User: CSR1
Role: CSR

User: CSR2
Role: CSR

Case Records		Record Visibility by Filter when VP Service is the running user			
Case Number	Case Owner	All	My	My Subordinates	My Team
1	CEO	X			
2	VP Service	X	X		X
3	Director Service	X		X	X
4	CSR1	X			X
5	CSR2	X			X

- My Queues: returns all records owned by the queues the running user is member of
- Check the Visibility Filters; make sure the selected filters are supported by your object. My Team is not supported on some standard objects. My, My Team and My Subordinates are not available for objects without an Owner field
- Default Visibility Filter: filter that will be used the first time the list is displayed; must be one of the selected filters
- If you need to build a list with one filter only (My Cases, My Team Leads), select the filter you need and the Visibility Filters combobox won't be displayed
- My Subordinates doesn't return records for users without a role

Files Settings

- Allowed Extensions:
 - If you want to restrict the extensions of the files that can be uploaded, enter a comma separated list of file extension. For example: txt,jpg
 - Leave blank if you don't want to have a restriction of file extensions
 - See [Files List Customization](#) for more details

Apex Settings

- Data Provider Class: name of the class of the data provider. See [Apex Data Sources](#)
- Row Key:
 - Field of the data source containing the unique identifier of a record returned by the data source
 - See [Apex Data Sources](#)

Step 2: Create Smart List Fields

Information

- Enter unique values in Label & Smart List Field Name; best practice: prefix the values with the name of the parent list

Field

- Smart List Definition: parent list
- Field Label:
 - Leave empty if you want to use the label of the field (SOQL and Files lists only)
 - Custom value: enter a value or enter `$Label.CustomLabelName` if you want to use a translatable label
- Field Name:
 - SOQL and Files lists:
 - API name of a field: can be a related field. Examples: `Account.Name` or `Branch__r.Code__c`
 - Fields are only displayed if the running user has read access on the field
 - Related fields are only displayed if the running user has read access on the related object and the related field
 - While SmartLists does not limit the depth of the relationships (`Contact.Account.Parent.Name`), adding related fields to lots of different objects has an impact on the performance of the component
 - RecordType predefined field: `RecordType.Name` is displayed in the list and a picklist of active record types is displayed in the Filters Panel
 - Owner predefined field: `Owner.Name` is displayed in the list and a lookup for selecting the owner is displayed in the Filters Panel
 - Apex Data Source: name of the field in the data source
 - Dynamic Style Field: API name of the text formula field returning the styling for a field
 - The returned value must be a list of styling tags separated by semicolons such as `tc:green;icn:utility:priority;icc:seagreen;icp:right`
 - Valid Style Tags:
 - `tc`: text color; valid values are
 - CSS color: `aliceblue`, `seagreen`,
 - rgb value: `rgb(240, 248, 255)`, `rgb(46, 139, 87)`
 - hex value: `#F0F8FF`, `#2E8B57`
 - `icn`: icon name; must be a Salesforce icon name; see List Icon in the Table section for more details
 - `icc`: icon color

- icp: icon position; valid values are:
 - left: icon displayed to the left of the field value
 - right: icon displayed to the right of the field value
 - hidevalue: icon displayed without the field value
- Lookup Subtitle Field: see [Lookups](#)

Display Type Settings

- Display Type:
 - Hyperlink to Detail
 - Select this value if you want to display a hyperlink to a record detail in the column
 - For Apex Data Sources, you need to enter the name of the field containing the Id of the related record in Hyperlink to Detail Id Field
 - File Preview (Files List only): add a hyperlink for opening the standard File Preview page. Because this page is not available in Digital Experiences sites, it is replaced by a hyperlink to the file detail page in a Digital Experience context
 - Currency Converted: in Multi-Currency orgs, display the value of a currency field converted to the currency of the running user
 - Currency Formatted: in Multi-Currency orgs, display the value of a currency field for the currency of the record and optionally converted to the currency of the running user if the currency of the record and the running are different
 - URL with Label
 - Select this value if you want to display a clickable label that navigates to a URL specified in the data source
 - The field containing the value of the URL must be specified in 'URL with Label: Value'
 - If you don't need to create a field definition for this field
 - If the value of URL Value is empty on a record, the label is not displayed
 - The target of the URL is '_blank' by default. You can optionally specify another target in 'URL with Label: Target'. The '_self' target can only be used on Salesforce URLs
 - Other values: ignored for SOQL and Files list types; required for Apex Data Source

- Hyperlink Id Field (used with Hyperlink to Detail):
 - Field of the list record containing the Id of the target record. Example: For a hyperlink to Account on Case, Hyperlink Id Field = AccountId
 - SOQL and Files lists
 - If left blank, SmartLists will automatically determine the value for you:
 - Base field of the list (Case Number for a list of Cases): Id of the record (Case.Id)
 - Related Field (Account.Name for a list of Cases): Id of the related record (Case.AccountId)
 - If Hyperlink to Detail is set on a formula field, you will need to populate this value because SmartLists cannot determine a value. Example for a formula displaying the name and city of the account of a case: AccountId
 - Apex lists: must be populated with the field of the data source containing the id of the target record. See [Apex Data Source for Record Detail Page](#)
 - You don't need to create a field definition for the field specified in Hyperlink Id Field
- URL with Label: Value and URL with Label: Target: see Display Type
- Lookup Subtitle field: for lookup fields, value of the subtitle field
- Lookup SOQL Filter: for lookup fields, SOQL filter used for filtering the records returned by the lookup

List Settings

- Display in List: if checked, the field is displayed in the list.
- Display Position in List: if the value is 1, the field will be displayed to the right of the field with value 0
- Sortable in List & Default Sort Field: see [Sorting Customization](#)
- Note: If you want to add HTML formatted formula fields, make sure that the height of the content is 19 pixels or lower. Otherwise, it will be truncated. Column wrapping is not available for this data type

Filters Settings

- Display in Filters Panel & Display Position in Filters Panel: see [Search Customization](#)
- Lookup in Filters Panel: see [Search Customization](#)

Table Settings

- Column Width: specify the initial width of the column in pixels; or leave blank if you want the system to automatically determine the width of the column
- Field Alignment: alignment of the value in the cell
- Inline Edit: the field is editable in the list with the following restrictions:
 - The running user has edit right on the field
 - The datatype is editable
 - The field is on the base object
 - The field is directly related to a parent of the base object
 - On Case, Contact.Name is editable because it is related to a parent of the base object
 - On Case, Contact.Account.Name is not editable because it is related to a grandparent of the base object
 - The dynamic style of edited field is only updated after the record is saved
 - Editing several rows at a time is partly supported for picklist and lookup types:

	Name	Account Name	Boolean
1	<input checked="" type="checkbox"/> Record1	<input type="checkbox"/> Burlington Textiles Corp of America	
2	<input checked="" type="checkbox"/> Record2	<input type="checkbox"/> Update 2 selected items	
3	<input type="checkbox"/> Record3		

Cancel Apply

Because of a Salesforce bug, checking Update x selected items before selecting a value will make the edit form disappear

- Wrap Text: see [Wrap Mode Customization](#)

Tiles Settings

- Display in Tile Header: if checked, the field is displayed in the tile header, otherwise, it is displayed in the tile. Several fields can be added to the header; they will be added based on the order specified in Display Position in List

Example: Tile Header with Lead Name and Lead Status



- Display as Badge: if checked, the field is displayed in the tile header as a badge with a dynamic styling
- Badge Style Field: Formula field returning the style of the badge for the current record
 - the formula field must return a string formatted as follows: bc:valid color; tc:valid color
See https://developer.mozilla.org/en-US/docs/Web/CSS/color_value for valid color codes
 - bc is used for specifying the background color of the badge
 - tc is used for specifying the color of the text of the badge
 - Example for Lead Rating dynamic styling: IF(ISPICKVAL(Rating, "Hot"),"bc:#2e844a;tc:white", IF(ISPICKVAL(Rating, "Warm"), "bc:#feca39;tc:black", "bc:#2172d5;tc:white"))

Files Settings

- Editable in File Edit Form:
 - –None--: the field is not displayed in the File Edit Form
 - Editable and Required: the field is displayed as required in the File Edit Form even if it is not required at the database level
 - Editable: the field is in the File Edit Form. It will be marked as required in the form if it is required at the database level
 - Note: Checkbox fields cannot be required in the File Edit Form

Note: you can create fields that are not displayed in the list and in the Filters if you need them in the flow actions

Step 3 - Optional: Create Smart List Filters

If you create only one filter, it will be used as default and the Filters combobox won't be displayed

- Enter unique values in Label & Smart List Filter Name; best practice: prefix the values with the name of the parent list
- Smart List Definition: parent list
- Filter Label: enter a value or enter \$Label.CustomLabelName if you want to use a translatable label
- Default Filter: check if you want this filter to be used when the list is displayed for the first time
- SOQL Filter: SOQL expression of the filter; for SOQL and Files lists only.
 - Example of filters for a list of cases:
 - Status = 'Open'
 - RecordType.DeveloperName = 'MyRt'
 - CreatedById = USERID. The variable USERID return the Id of the running user
 - SOQL [date functions](#) and [date literals](#) are available

Step 4 - Optional: Create Smart List Actions

- Enter unique values in Label & Smart List Action Name; best practice: prefix the values with the name of the parent list
- Smart List Definition: parent list
- Action Label: enter a value or enter \$Label.CustomLabelName if you want to use a translatable label
- Type:
 - List Action: action executed on the rows selected in the list; the button of this action is displayed at the top of the list when rows are selected
 - Row Action: action executed on a singled row; the menu item of this action is displayed in the row actions menu
- Display Position: display position in the list of buttons or in the row actions menu. If you enter 1; the action will be displayed after the action with 0
- Refresh After Execution: determine if the list of the row must be refreshed after the execution of the action
- Category: type of flow Autolaunched or Screenflow
- Custom Permission: if populated, the action is available if the running user has this custom permission; API name of the custom permission

- Flow Name: API name of the flow
 - See [Autolaunched Flow Example](#)
 - Your flow must have the following variables:

API Name	Required	Resource Type	Data Type	Object	Allow multiple values	Availability outside the flow	Description
records	Yes	Variable	Record	Base object of the list	Checked	Available for input	List of records passed to the flow. Each record includes the Id field and all the list fields (see List Action Availability for records passed) If your flow need a field that you don't want to display in the list or in the Filters Panel, create a Field Definition with Display in List = false and Display in Filters = No
parentId	Yes	Variable	Text	N/A	Unchecked	Available for input	For child lists, id of the parent record; null otherwise
successMsg	No	Variable	Text	N/A	Unchecked	Available for output	Message to display if the action is successful No message will be displayed if no value is returned
errorMsg	No	Variable	Text	N/A	Unchecked	Available for output	Message to display if the action fails No message will be displayed if no value is returned Note: successMsg will be displayed instead of errorMsg if it has a value

- List Action Availability:

Value	Available when	Records passed to the flow
When Records are Selected	One or more records are selected in the list Example: close selected cases	Selected records
When Records are Loaded	Records are loaded in the list Example: add the contacts returned by a search to a campaign	Loaded records
Always	Available all the time Example: custom New record action	Loaded records or empty array if no records loaded

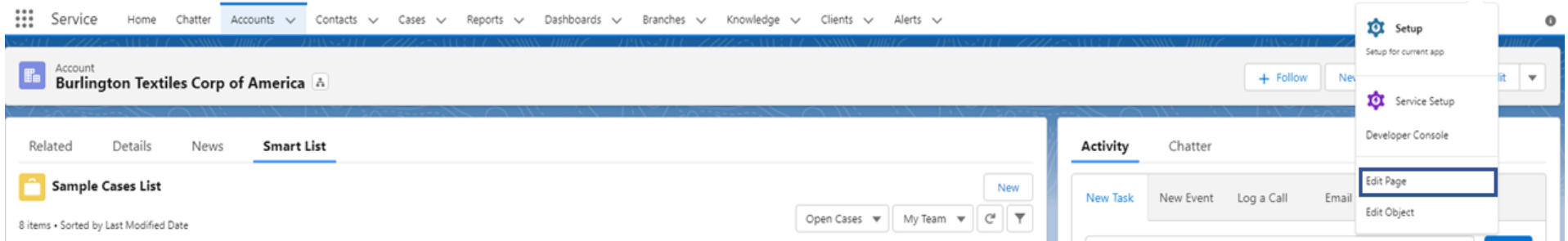
- Screenflow Modal Height: height in pixels of the modal dialog used for displaying the screen flow; not needed for auto-launched flows; if left blank the height of the modal is based on the height of the flow screen
- Screenflow: Show Label in Modal Header: if checked, the label of the action is displayed in the header of the modal dialog. Otherwise, the header is not displayed

Step 5 – Add the list to a target Page

- Prerequisite: make sure, the Apex Class 'smartLists.SmartListController' is accessible by your user

Lightning App Builder Page

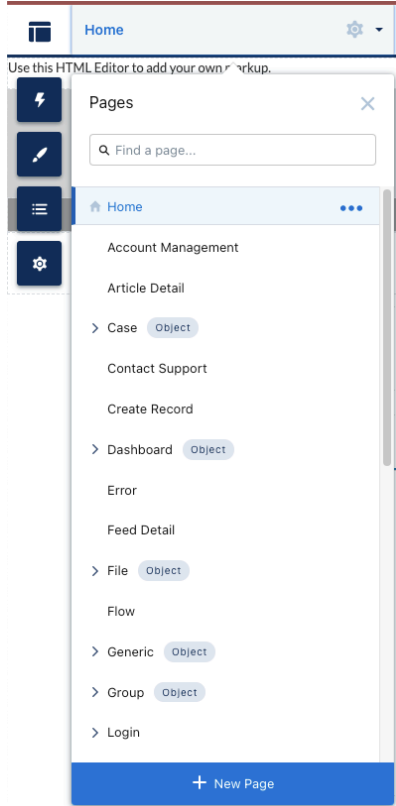
- Navigate to the page where you want to add the list
- Edit the App Builder page



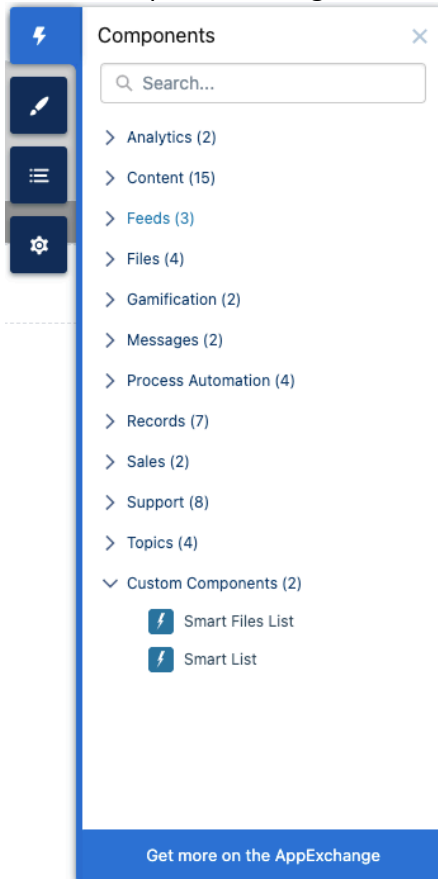
- In the Components widget on the left, select the Smart List / Smart Files List component in Custom – Managed
- Drag the component on the page
- Enter the parameters of the component
 - List Definition Name: name of the List Definition
- Activate the page if needed and save it

Digital Experience Page

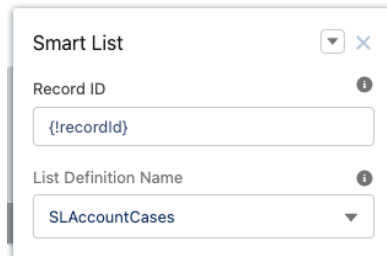
- Open your site in Experience Builder
- Select the page where you want to add your list



- In the Components widget on the left, select the Smart List / Smart Files List component in Custom Components



- Drag the component on the page
- Enter the parameters of the component



- For a list of child records, Record ID must be populated with {!recordId}
- List Definition Name: name of the List Definition
- Preview your changes
- Click Publish

Screenflow

- Consideration for Screenflows
 - Hyperlinks to Detail are not available in Screenflows because you need to end the flow before navigating to another screen; as a workaround, you can configure your list for allowing the selection of 1 record at a time (Max Row Selected > 1 on List Definition) and build a custom [Custom Flow Action](#) that will close the flow and navigate to the record after a click on Next
 - Screenflows actions are not available in Screenflows because of a Salesforce limitation
- Create a Screenflow
- If row selection is allowed in your list, create a variable for receiving the record(s) selected in the list if you need to use the selected records in your flow

- Data Type: Record

- Object: SObject of the list

- Allow multiple values: checked if the list is configured for selecting more than 1 records (Max Row Selected > 1 on List Definition):

selectedLead 

* Data Type 

Record ▼

Allow multiple values (collection) 

* Object

Lead

Availability Outside the Flow

Available for input

Available for output

- Add the Smart List / Smart Files List component to a screen

The screenshot shows the Salesforce Flow Builder interface. On the left, a canvas titled 'SLScreenFlow' contains a 'Smart List' component. Below the component is a 'Pause' button and 'Previous' and 'Finish' navigation buttons. On the right, the configuration sidebar for the 'Smart List' component is visible, showing the following settings:

- * API Name: SmartListLeads
- * Input Type for Selected Records parameter: Lead
- * INTERNAL - Must be set to `{!$GlobalConstant.True}`: `{!$GlobalConstant.True}`
- List Definition Name: SLLeads
- Minimum Row Selected: 1
- Record ID: Enter value or search resources...
- Set Component Visibility: (expanded)
- Advanced:
 - Manually assign variables

- Enter the properties of the component
 - Input Type: object type of the list – Not available for Smart Files Lists
 - INTERNAL: must be set to `{!$GlobalConstant.True}`
 - List Definition Name: API name of the list definition
 - Minimum Row Selected: enter a value if you want to make sure users select at least a specific number of rows before moving to the next screen
 - Record ID: flow variable containing the parent id of the list; leave blank if the list is not related to a parent record
 - Advanced / Manually assign variables:
 - If unchecked (required for reactive flows), use the variables of the component:
 - Selected Record: selected record for single selection lists. Data type: Record of SObject of the list
 - Selected Records: selected records for multi selection lists. Data type: collection of Record of SObject of the list
 - Selected Records Count: number of records selected in the list. Data type: Number
 - If checked, assign the output values to your own variables

Lookups

Filters ×

Case Number ×

Contact ×

edge Q ×

Search Results

- 🔍 Rose Gonzalez
SVP, Procurement
- 🔍 Sean Forbes
CFO

New

Working

Name Field of the record (Name, CaseNumber...)

Lookup Subtitle

Lookups are used for retrieving a related record with inline edit and optionally in the Filters Panels. The related record is searched on the Name field (Account Name, Case Number) and on the Lookup Subtitle Field if provided

The list of matching results contains the name and optionally the subtitle

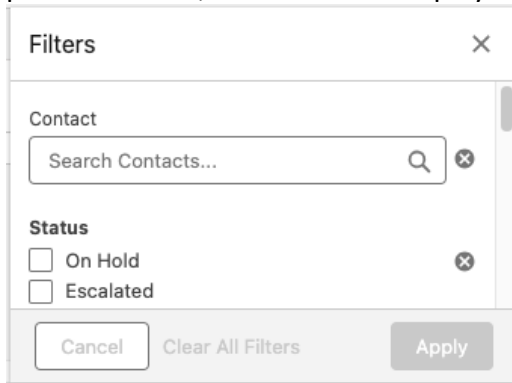
Lookup Subtitle Field: field of the related record that is displayed in the search results for an easier identification of the records

- subtitle is optional
- must have the data type Text, Email, Phone or Picklist
- not displayed if the running user does not have read access to the field

Search Customization

- List Definition
 - Filters Panel position and behavior:
 - Filters Panel Layout:
 - Left – XYZ: the panel is displayed on the left of the list
 - Right – XYZ: the panel is displayed on the right of the list
 - On-Demand: the panel is displayed when the Filters button is clicked
 - All the Time: the panel is displayed all the time and cannot be closed
 - If you want to hide the Filters Panel, mark all your Fields as not Visible in Filters Panel. See below
 - Filters Panel Max Height:
 - If a value is not specified, all the filters are displayed in the list and users may have to scroll to view all of them and click the panel buttons

- Otherwise, the height of the panel is set to the specified value. If the height in pixels of all the filters is higher than the provided value, the filters are displayed in a scrollable window



The image shows a 'Filters' dialog box with a scrollable content area. The dialog has a title bar with 'Filters' and a close button. The content area is divided into sections: 'Contact' with a search input field, and 'Status' with two checkboxes. The 'Apply' button is highlighted.

Filters

Contact

Search Contacts...

Status

On Hold

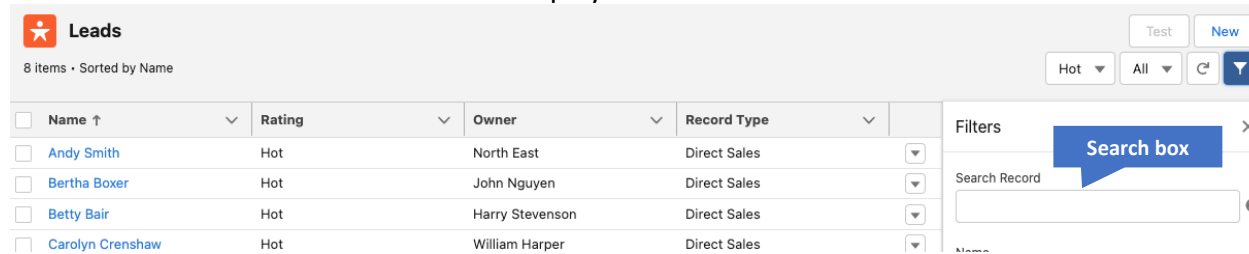
Escalated

Cancel Clear All Filters Apply

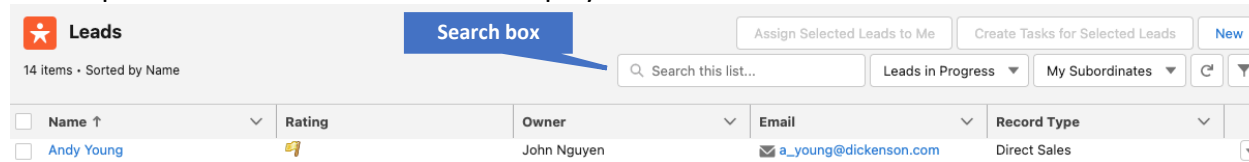
○ SOSL Search:

- The SOSL search box visibility and location are controlled by the value set in Show SOSL Search:

- In Filters Panel: the SOSL search box is displayed in the Filters Panel



- In Component: the SOSL search box is displayed above the list

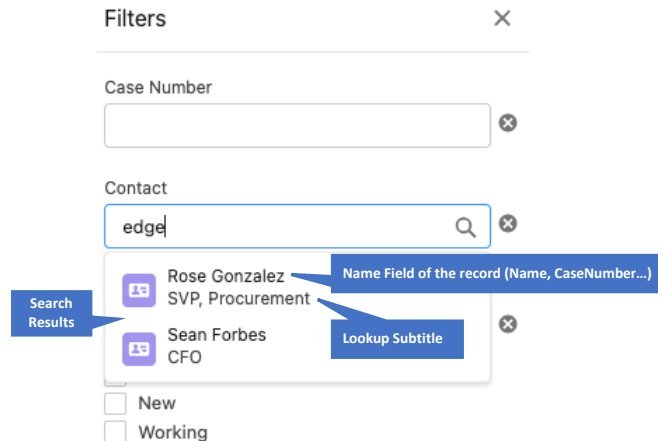


- Not Displayed

- SOSL search is only searching the text fields (Email, Phone, String, Text Area, Url); related fields or fields with other data types are not searchable
- The SOSL search box is not displayed if the list has no searchable fields
- Some objects, such as Opportunity Line Item or Campaign Member, are not searchable with SOSL. The SOSL search box is not displayed for these objects

- Field Definition

- Display in Filters Panel: select No if you don't want to display the field in the Filters Panel; fields which are not searchable because of their datatype or Shield Probabilistic Encryption are not displayed in the Filters Panel even if Display in Filters is selected
- Display Position in Filters Panel: display position in the Filters Panel. If you enter 1; the field will be displayed after the field with 0
- Lookups in Filters Panel: if checked, the search is

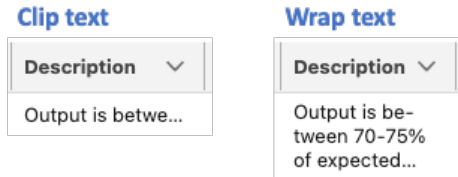


- Lookups are used for filtering on record Owner and related records in the Filters Panel
- In the search results, the name field of the record (Name, CaseNumber...) as well as an optional subtitle field are displayed
- Lookup in Filters Panel: if checked, the field is displayed as a lookup; otherwise it is displayed as a text field
- Lookup Subtitle Field: field of the related record that is displayed in the search results for an easier identification of the records
 - subtitle is optional
 - must have the data type Text, Email, Phone or Picklist
 - not displayed if the running user does not have read access to the field
- Search on text fields with Shield Deterministic Encryption:
 - Encrypted text fields are searched based on the exact value provided in the panel (= operator)
 - Non-encrypted text fields are searched with the LIKE operator
- Search on text fields of External Objects:
 - The LIKE operator is not available on External Objects. Text fields are searched based on the exact value provided in the panel (= operator)
- If you don't want to allow search in your list, make sure that Visible in Filters Panel is set to No on all the fields

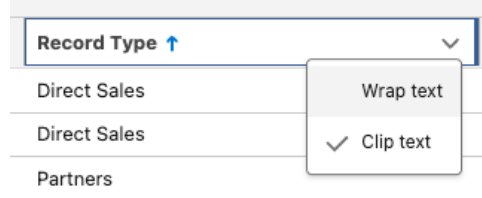
Wrap text mode

By default, values are displayed in Clip text mode. Values which don't fit the width of the cell or have several lines are truncated on 1 line terminated by ellipsis.

Wrap text mode can be used to display the values on several lines.



Users can switch between modes by using the corresponding column actions in the header of the column



The Wrap text mode of a column is controlled by the following parameters:

- List Definition
 - Wrap Text Max Lines: number of lines before wrapping the content of a cell when the column is in Wrap text mode; if you enter 3 and the cell contains 4 lines, 3 lines will be displayed in the cell and 3 dots will be added at the end of the third line to indicate there are more lines in the cell
- Field Definition
 - Wrap Text: check if you want Wrap text to be the default mode for the column

Sorting Customization

- List Definition
 - Default Sort Direction: sort direction used for the Default Sort Field the first time the list is displayed
- Field Definition
 - Sortable in List: select No if you want to disable the sort on this field to avoid performance issues
 - Default Sort Field: check if the records must be sorted on this field the first time the list is displayed

Files List Customization

Standard Actions

Access to standard actions is controlled by custom permissions that can be added to a profile or a permission set:

Label	API Name	Description
Don't check file extension	SmartFilesList_Don_t_check_file_extension	Bypass the file extension check for Upload File and Upload New Version actions
Download Files	SmartFilesList_Download_Files	Control access to File Download action
Edit File Details	SmartFilesList_Edit_File_Details	Control access to Edit File Details action
Preview Files	SmartFilesList_Preview_Files	Control access to Preview File action
Upload Files	SmartFilesList_Upload_Files	Control access to Upload File action
Upload New Version	SmartFilesList_Upload_New_Version	Control access to Upload New Version action
View File Details	SmartFilesList_View_File_Details	Control access to View File Details action

Note: some standard actions are not available for all targets. See [Features by List Type and Targets](#) for more details

Restrict Extensions of Uploaded Files

It is possible to restrict the extensions of the files uploaded with the File Upload and File Upload new Version actions.

The valid extensions are defined on the list definition in Allowed Extensions as a comma separated list of extensions such as jpg,txt.

The control of the extensions is only happening if a value has been entered in Allowed Extensions and the running user **does not have** the custom permission Don't check file extension

File Edit Form

It is possible to create a File Edit Form which is displayed when new files are uploaded or when Edit File Details is selected.

The fields are added to the form by entering a value in 'Editable in File Edit Form' of the Field Definition:

- Editable and Required: the field is displayed in the form as required if the running user has access to the field
- Editable: the field is displayed in the form if the running user has access to the field. The field is marked as required if it is required at the object level

The fields are displayed in the form based on the order specified in Display Position in List

Considerations for object, and field access

- An error message is displayed if the running user doesn't have read access to the object of the list
- List fields are not displayed in the list if the running user doesn't have read access to the field
- An error message is displayed if none of the list fields are visible by the running user

Considerations for standard record actions

- New, Edit and Delete actions are only available if the running user has the corresponding access rights on the object
- The New/Edit form is using the page layout assigned to the running user
- New and Edit are not supported for Person Account, Event and Task

External Object support

Restrictions and Limitations

- Record actions are not supported
- Lookups in the Filters panel are not supported
- Visibility filters are ignored
- Related fields are not sortable
- Text fields are searched with the = operator instead of LIKE

Configuration of relationships

- Child relationships with a standard or custom object are configured as follows:
 - An Indirect Lookup field must be added to the external object:
 - Related to: parent standard or custom object
 - Target Field: field of the parent object containing the Id of the parent record available in the external object
 - External Column Name: field of the external object containing the id of the parent record
 - Example for a relationship to Account on Order (Order__x):
 - Account field ERP_Account_Number__c contains the ERP account number
 - External object field Account_Number__c contains the ERP account number and is mapped to External Column Name accountNumber
 - Indirect Lookup field Account__c on Order__x:
 - Related To: Account
 - Target Field: ERP_Account_Number__c
 - External Column Name: accountNumber

- An indirect relationship must be configured as follows on the List Definition:
 - Parent Id Field: Account__c (field on child external object)
 - Indirect Relationship: Parent Key Field: ERP_Account_Number__c (field on Account)
- Child relationships between related external objects are configured as follows:
 - An External Lookup field must be added to the child external object:
 - Related to: parent external object
 - External Column Name: field of the child external object containing the id of the parent record
 - Example for a relationship to Order (Order__x) from Order Line (Order_Line__x):
 - Order__x field Order_Number__c contains the order number
 - Order_Line__x field Order_Number__c contains the order number and is mapped to orderNumber
 - External Lookup field Order__c on Order_Line__x:
 - Related To: Order__x
 - External Column Name: orderNumber
 - An indirect relationship must be configured as follows on the List Definition:
 - Parent Id Field: Order_Number__c (field on child external object)
 - Indirect Relationship: Parent Key Field: ExternalId

Features by List Type and Targets

Availability by Target

Feature	Targets		
	Lightning App Builder pages	Digital Experience site pages	Screen Flow
List Type			
SOQL List	Home, App, and Record pages	All pages	Everywhere
Files List	Record page	Record detail page	Everywhere
Apex Data Source	Home, App, and Record pages	All pages	Everywhere
Standard Record Actions			
New	Available	Available	Available
Edit	Available	Available	Available
Delete	Available	Available	Available
Standard File Actions			
File Upload	Available	Available	Available
File Download	Available	Available	Available
File Preview	Available	Automatically replaced by View File Details	Not available
View File Details	Available	Available	Not available
Edit File Details	Available	Available	Available
Upload New Version	Available	Available	Available
Other Standard Actions			
Export to CSV	Available	Available	Available
Custom Actions			
Screenflow	Available	Available	Not available
Auto-launched Flow	Available	Available	Available

Features by List Type

Feature	List Types		
	SOQL List	Files List	Apex Data Source
Inline Edit	Available	Available	Not Available
SOSL Search	All text fields of the records	Content of the files	All text fields of the records
Filters Panel fields	All filterable fields of the base object with Display in Filters set to Yes	All filterable fields of Content Version with Display in Filters set to Yes	All fields with Display in Filters set to Yes
Custom Filters	Available	Available	Available
Export to CSV	Available	Available	Available
Visibility Filters			
All	Available	Available	Available
My	if supported by object	Available	Available
My Subordinates	if supported by object	Available	Available
My Queues	If supported by object	Available	Available
My Team	if supported by object	Not Available	Available

Note for Apex Data Source: the support for the features must be implemented in the data provider

Programmatic Updates of the Smart List with Lightning Messaging Service

Apex Data Source for Record Detail Page

Message must be published to the 'SmartList__c' Message Channel

Apex Data Source for Record Detail Page

Type of Update	Message Example	Parameters
Refresh list	<pre>{"list": "*", "action": {"type": "REFRESH"}}</pre>	list: '*' for refreshing all lists or Smart List Definition Name of the list to refresh (example: Sample_Leads_List) action.type: REFRESH
Set custom filter	<pre>{"list": " Sample_Leads_List", "action": {"type": "FILTER", "filter": "Leads_Open_Leads"}}</pre>	action.type: FILTER action.filter: Smart List Filter Name of the filter (example: Leads_Open_Leads)
Set SOQL Scope	<pre>{"list": " Sample_Leads_List", "action": {"type": "SCOPE", "scope": "mine"}}</pre>	action.type: SCOPE action.scope: <ul style="list-style-type: none">- everything: 'All' scope- mine: 'My' scope- team: 'My Team' scope- subordinates: 'My Subordinates' scope- queues: 'My Queues' scope

Sample LMS Publisher

IMSPublisher.html

```
<lightning-card title="LMS Publisher">
  <div class="slds-var-p-around_medium lgc-bg">
    <lightning-input type="text"
      label="** for targetting all the Smart Lists of the page or Developer Name of one of the Smart Lists of the page"
      value={target} onchange={handleTargetChange}></lightning-input>
  </div>
  <lightning-tabset>
    <lightning-tab label="Refresh List">
      <div class="slds-var-p-around_medium">
        <lightning-button label="Publish" onclick={handleRefresh}></lightning-button>
      </div>
    </lightning-tab>
    <lightning-tab label="Select Filter">
      <lightning-layout horizontal-align="spread">
        <lightning-layout-item size="10" padding="around-small">
          <lightning-input type="text"
            label="Filter Developer Name" onchange={handleFilterChange} variant="label-inline"></lightning-input>
        </lightning-layout-item>
        <lightning-layout-item size="2" padding="around-small">
          <lightning-button label="Publish" onclick={handleSelectFilter}></lightning-button>
        </lightning-layout-item>
      </lightning-layout>
    </lightning-tab>
    <lightning-tab label="Select Scope">
      <lightning-layout horizontal-align="spread" >
        <lightning-layout-item size="5" padding="around-small">
          <lightning-combobox label="Scope" options={scopeOptions} value={scope} onchange={handleScopeChange} variant="label-inline"></lightning-combobox>
        </lightning-layout-item>
        <lightning-layout-item size="7" padding="around-small">
          <lightning-button label="Publish" onclick={handleSelectScope}></lightning-button>
        </lightning-layout-item>
      </lightning-layout>
    </lightning-tab>
  </lightning-tabset>
```

```
<p slot="footer">
  Published Message:&nbsp;<lightning-formatted-text value={message}></lightning-formatted-text>
</p>
</lightning-card>
</template>
```

IMSPublisher.js

```
import { LightningElement, wire } from 'lwc';

// Import message service features required for publishing and the message channel
import { publish, MessageContext } from 'lightning/messageService';
import SMARTLIST_CHANNEL from '@salesforce/messageChannel/SmartList__c';

export default class LmsClient extends LightningElement {
  @wire(MessageContext)
  messageContext;

  // COMPONENT VARIABLES
  // Target
  _target = '*';
  get target() {
    return this._target;
  }
  // List Filter
  _filter;
  get filter() {
    return this._filter;
  }
  // Scope Options
  scopeOptions =
    [
      { label: 'All', value: 'everything' },
      { label: 'My', value: 'mine' },
      { label: 'My Team', value: 'team' },
      { label: 'My Subordinates', value: 'subordinates' },
      { label: 'My Queues', value: 'queues' },
    ];
};
```

```
_scope = 'everything';
get scope() {
  return this._scope;
}

// Published message
_message;
get message() {
  return JSON.stringify(this._message);
}

// TARGET LIST
// Target List Value Change
handleTargetChange(event) {
  this._target = event.detail.value;
}

// REFRESH LIST
// Publish Refresh Event
handleRefresh() {
  this._message = { list: this.target, action: {type: 'REFRESH' } };
  publish(this.messageContext, SMARTLIST_CHANNEL, this._message);
}

// FILTER
handleFilterChange(event) {
  this._filter = event.detail.value;
}
handleSelectFilter() {
  this._message = { list: this.target, action: {type: 'FILTER', filter: this.filter } };
  publish(this.messageContext, SMARTLIST_CHANNEL, this._message);
}

// SCOPE
handleScopeChange(event) {
  console.log('scopeChange ' + JSON.stringify(event.detail));
  this._scope = event.detail.value;
}
```

```
}  
handleSelectScope() {  
  this._message = { list: this.target, action: {type: 'SCOPE', scope: this.scope }};  
  publish(this.messageContext, SMARTLIST_CHANNEL, this._message);  
}  
}
```

Apex Data Sources

Apex Data Source for Record Detail Page

Step 1: Create an Apex class with the following code. Make sure your user has access to the class:

```
global with sharing class ApexDataSourceWithParent implements smartLists.SmartListApexSourceInterface2 {
    // Base query used by getPage and getRecords
    Static String baseQuery = 'SELECT AccountId, StageName, Sum(Amount) OpptiesSum, Count(Id) OpptiesCount, Owner.Name Owner FROM Opportunity';
    // Group By clause used by getPage and getRecords
    Static String groupByClause = ' GROUP BY StageName, AccountId, Owner.Name';

    // Get a list page
    public List<Map<String, Object>> getPage(smartLists.SmartListApexSourceGetPage parms) {
        String query = baseQuery;
        // Add visibility filter to the query
        query += ' USING SCOPE ' + parms.getScope();
        // Add relationship with parent record to where clause
        String whereClause = parms.getParentIdField() + '=' + parms.getParentId() + '\\";
        // If predefined filter defined for the list, add the filter to the where clause
        whereClause += String.isEmpty(parms.getFilter()) ? "" : ' AND (' + parms.getFilter() + ')';
        String havingClause = "";
        // If values entered in Filters Panel, add them to the where and having clauses
        if (parms.getFilterEntries() != null && parms.getFilterEntries().size() > 0) {
            List<smartLists.SmartListController.FilterEntry> whereEntries = new List<smartLists.SmartListController.FilterEntry>();
            List<smartLists.SmartListController.FilterEntry> havingEntries = new List<smartLists.SmartListController.FilterEntry>();
            for (smartLists.SmartListController.FilterEntry fe : parms.getFilterEntries()) {
                if (fe.fieldName == 'Sum(Amount)')
                    havingEntries.add(fe);
                else if (fe.fieldName == 'Count(Id)')
                    havingEntries.add(fe);
                else
                    whereEntries.add(fe);
            }
            String filtersWidget = smartLists.SmartListController.buildFilter(whereEntries);
            whereClause += String.isEmpty(filtersWidget) ? "" : (String.isEmpty(whereClause) ? filtersWidget : ' AND (' + filtersWidget + ')');
            havingClause = smartLists.SmartListController.buildFilter(havingEntries);
        }
        // Add where clause to query
        query += String.isEmpty(whereClause) ? "" : ' WHERE ' + whereClause;
        // Add group by to query
        query += groupByClause;
        // Add having clause to query
        query += String.isEmpty(havingClause) ? "" : ' HAVING ' + havingClause;
        // If sort field passed, add sort parameters to query
        query += String.isEmpty(parms.getSortField()) ? "" : ' ORDER BY ' + parms.getSortField() + ' ' + parms.getSortDirection();
        // Add paging parameters to query
    }
}
```

```

query += ' LIMIT ' + parms.getPageSize() + ' OFFSET ' + parms.getOffset();
// Retrieve the records
return getRecords(query);
}

// Get a list record
public List<Map<String, Object>> getRecord(String id) {
    String query = baseQuery;
    // Add where clause to query
    query += ' WHERE AccountId = \'' + id + '\'';
    // Add group by to query
    query += groupByClause;
    // Retrieve the records
    return getRecords(query);
}

// Query the database and format the records for Smart Lists
private List<Map<String, Object>> getRecords(String query) {
    System.debug('Query ' + query );
    // Query the database
    AggregateResult[] oppties = Database.query(query);
    // Parse the returned records and format them for Smart List
    System.debug('Oppties ' + oppties);
    List<Map<String, Object>> results = new List<Map<String, Object>>();
    for (AggregateResult oppty : oppties) {
        Map<String, Object> record = new Map<String, Object>();
        record.put('RowKey', (String)oppty.get('AccountId') + (String)oppty.get('StageName') + (String)oppty.get('Owner')); // Note: Required field for identifying a unique record
        record.put('Id', oppty.get('AccountId')); // Note: A field called Id is required in the list for using flow actions
        record.put('StageName', oppty.get('StageName'));
        record.put('Sum(Amount)', oppty.get('OpptiesSum'));
        record.put('Count(Id)', oppty.get('OpptiesCount'));
        record.put('Owner.Name', oppty.get('Owner'));
        results.add(record);
    }
    return results;
}
}

```

Step 2: Create a list definition

▼ List Settings

Data Source Type	Apex Data Source	Display Mode	Table
List Label	Sample Apex List with Parent	List Icon	standard:opportunity
Maximum Number of Records	2,000	Number of Records per Page	10
Selectable Rows	<input checked="" type="checkbox"/>	Max Row Selected	2,000
Display Filters Panel All the Time	<input type="checkbox"/>	Show SOSL Search	Not Displayed
		Default Sort Direction	asc

▼ Child Lists - Relationship Settings

Parent Id Field (Not for Files Lists)	AccountId	Indirect Relationship: Parent Key Field	
---------------------------------------	-----------	---	--

▼ Table Settings

Show Table Header	<input checked="" type="checkbox"/>	Wrap Text Max Lines	
Show Row Number Column	<input type="checkbox"/>	Row Number Start	0

► Tiles Settings

▼ SOQL with/without Sharing & Apex Settings

Object		Enable New Record Action	<input type="checkbox"/>
Enable all Record Actions	<input type="checkbox"/>	Enable Delete Record Action	<input type="checkbox"/>
Enable Edit Record Action	<input type="checkbox"/>	Visibility Filter - All	<input checked="" type="checkbox"/>
Default Visibility Filter	All	Visibility Filter - My Team	<input checked="" type="checkbox"/>
Visibility Filter - My	<input checked="" type="checkbox"/>	Visibility Filter - My Subordinates	<input type="checkbox"/>

► Files Settings

▼ Apex Settings

Data Provider Class	ApexDataSourceWithParent	Row Key	RowKey
---------------------	--------------------------	---------	--------

Notes:

- Parent Id Field is used for passing the parent id to the Apex Class
- Data Provider Class is the name of the class created at Step 1
- RowKey is the name of the field populated in the Apex Data Source. It must contain a unique row identifier otherwise, a Javascript error may happen when the list is refreshed or updated

Step 3: Create the fields

Field				List Settings				Filters Settings	
Field Label	Field Name	Display Type	Hyperlink to Detail Id Field	Display in List	Display Position in List	Sortable in List	Default Sort Field	Display in Filters Panel	Display Position in Filters Panel
Stage	StageName	String		TRUE	0	Yes	FALSE	Yes	0
Number of Opportunities	Count(Id)	Integer		TRUE	1	Yes	FALSE	Yes	1
Total Amount	Sum(Amount)	Currency		TRUE	2	Yes	FALSE	Yes	2
Owner	Owner.Name	String		TRUE	3	Yes	FALSE	Yes	3

Note:

- Field Label and Display Type are required for an Apex Data Source

Step 4: Create the predefined filters

Filter Label	Default Filter	SOQL Filter
Won Opportunities	FALSE	IsClosed = true AND isWon = true
Opportunities in Progress	TRUE	IsClosed = false
Lost Opportunities	FALSE	IsClosed = true AND isWon = false

Step 5: Add the list to your Account Detail Page

Test Class

```
@isTest
public with sharing class ApexDataSourceWithParentTest {
    @isTest
    static void testGetPage() {
        // Create the test data
        Account acc = new Account(name = 'test');
        insert acc;
        Opportunity oppty1 = new Opportunity(name = 'Oppty1', AccountId = acc.Id, CloseDate = System.today(), StageName = 'Prospecting', Amount=100);
        Opportunity oppty2 = new Opportunity(name = 'Oppty2', AccountId = acc.Id, CloseDate = System.today(), StageName = 'Value Proposition', Amount=100);
        Opportunity oppty3 = new Opportunity(name = 'Oppty3', AccountId = acc.Id, CloseDate = System.today(), StageName = 'Closed Won', Amount=100);
        List<Opportunity> oppties = new List<Opportunity>{oppty1, oppty2, oppty3};
        insert oppties;
        // Create values from Filters Panel
        smartLists.SmartListController.FilterEntry fe1 = new smartLists.SmartListController.FilterEntry();
        fe1.fieldName = 'StageName';
        fe1.operator='LIKE';
        fe1.values = new String[]{'Prospecting' };
        fe1.type='STRING';
        smartLists.SmartListController.FilterEntry fe2 = new smartLists.SmartListController.FilterEntry();
        fe2.fieldName = 'Sum(Amount)';
        fe2.operator='>=';
        fe2.values = new String[]{'100'};
        fe2.type='CURRENCY';
        smartLists.SmartListController.FilterEntry fe3 = new smartLists.SmartListController.FilterEntry();
        fe3.fieldName = 'Count(Id)';
        fe3.operator='>=';
        fe3.values = new String[]{'1'};
        fe3.type='INTEGER';
        List<smartLists.SmartListController.FilterEntry> fes = new List<smartLists.SmartListController.FilterEntry>{fe1, fe2, fe3};
        Test.startTest();
        // Define getPage parameters
        smartLists.SmartListApexSourceGetPage getPageParms =
            new
smartLists.SmartListApexSourceGetPage().withScope('everything').withFilter('').withFilterEntries(fes).withParentIdField('AccountId').withParentId(acc.Id).withSortField('StageName').withSortDirecti
on('asc').withOffset(0).withPageSize(10);
        ApexDataSourceWithParent ds = new ApexDataSourceWithParent();
        // Invoke getPage
        List<Object> result = ds.getPage(getPageParms);
        Test.stopTest();
        System.assertEquals(1, result.size());
    }

    @isTest
    static void testGetRecord(){
        // Create the test data
        Account acc1 = new Account(name = 'test1');
```

```
Account acc2 = new Account(name = 'test2');
List<Account> accs = new List<Account>{acc1, acc2};
insert accs;
Opportunity oppty1 = new Opportunity(name = 'Oppty1', AccountId = acc1.Id, CloseDate = System.today(), StageName = 'Prospecting');
Opportunity oppty2 = new Opportunity(name = 'Oppty2', AccountId = acc2.Id, CloseDate = System.today(), StageName = 'Value Proposition');
List<Opportunity> oppties = new List<Opportunity>{oppty1, oppty2};
insert oppties;
Test.startTest();
ApexDataSourceWithParent ds = new ApexDataSourceWithParent();
// Invoke getRecord
List<Object> result = ds.getRecord(acc1.Id);
Test.stopTest();
System.assertEquals(1, result.size());
}
}
```

Apex Data Source for Home Page or Custom Tab

Step 1: Create an Apex class with the following code. Make sure your user has access to the class:

```
global with sharing class ApexDataSource implements smartLists.SmartListApexSourceInterface2 {
    // Base query used by getPage and getRecords
    Static String baseQuery = 'SELECT AccountId, Account.Name AccountName, StageName, Sum(Amount) OpptiesSum, Count(Id) OpptiesCount, Owner.Name Owner FROM Opportunity';
    // Group by used by getPage and getRecords
    Static String groupByClause = ' GROUP BY StageName, AccountId, Account.Name, Owner.Name';

    // Get a list page
    public List<Map<String, Object>> getPage(smartLists.SmartListApexSourceGetPage parms) {
        String query = baseQuery;
        // Add visibility filter to the query
        query += ' USING SCOPE ' + parms.getScope();
        // If predefined filter defined for the list, add the filter to the where clause
        String whereClause = String.isEmpty(parms.getFilter()) ? '' : parms.getFilter();
        String havingClause = '';
        // If values entered in Filters Panel, add them to the where and having clauses
        if (parms.getFilterEntries() != null && parms.getFilterEntries().size() > 0) {
            List<smartLists.SmartListController.FilterEntry> whereEntries = new List<smartLists.SmartListController.FilterEntry>();
            List<smartLists.SmartListController.FilterEntry> havingEntries = new List<smartLists.SmartListController.FilterEntry>();
            for (smartLists.SmartListController.FilterEntry fe : parms.getFilterEntries()) {
                if (fe.fieldName == 'Sum(Amount)')
                    havingEntries.add(fe);
                else if (fe.fieldName == 'Count(Id)')
                    havingEntries.add(fe);
                else
                    whereEntries.add(fe);
            }
            String filtersWidget = smartLists.SmartListController.buildFilter(whereEntries);
            whereClause += String.isEmpty(filtersWidget) ? '' : (String.isEmpty(whereClause) ? filtersWidget : ' AND (' + filtersWidget + ')');
            havingClause = smartLists.SmartListController.buildFilter(havingEntries);
        }
        // Add where clause to query
        query += String.isEmpty(whereClause) ? '' : ' WHERE ' + whereClause;
        // Add group by clause to query
        query += groupByClause;
        // Add having clause to query
        query += String.isEmpty(havingClause) ? '' : ' HAVING ' + havingClause;
        // If sort field passed, add sort parameters to query
        query += String.isEmpty(parms.getSortField()) ? '' : ' ORDER BY ' + parms.getSortField() + ' ' + parms.getSortDirection();
        // Add paging parameters to query
        query += ' LIMIT ' + parms.getPageSize() + ' OFFSET ' + parms.getOffset();
        // Retrieve the records
        return getRecords(query);
    }

    // Get a list record
```

```

public List<Map<String, Object>> getRecord(String id) {
    // Add where clause to query
    String query = baseQuery + ' WHERE AccountId = \'' + id + '\'';
    // Add group by to query
    query += groupByClause;
    // Retrieve the records
    return getRecords(query);
}

// Query the database and format the records for Smart Lists
private List<Map<String, Object>> getRecords(String query) {
    List<Map<String, Object>> results = new List<Map<String, Object>>();
    System.debug('Query ' + query );
    // Query the database
    AggregateResult[] oppties = Database.query(query);
    // Parse the returned records and format them for Smart List
    for (AggregateResult oppty : oppties) {
        Map<String, Object> record = new Map<String, Object>();
        record.put('RowKey', (String)oppty.get('AccountId') + (String)oppty.get('StageName') + (String)oppty.get('Owner')); // Note: Required field for identifying a unique record
        record.put('Id', oppty.get('AccountId')); // Note: A field called Id is required in the list for using flow actions
        record.put('Account.Name', oppty.get('AccountName'));
        record.put('StageName', oppty.get('StageName'));
        record.put('Sum(Amount)', oppty.get('OpptiesSum'));
        record.put('Count(Id)', oppty.get('OpptiesCount'));
        record.put('Owner.Name', oppty.get('Owner'));
        results.add(record);
    }
    return results;
}
}

```

Step 2: Create a list definition

▼ List Settings

Data Source Type	Apex Data Source	Display Mode	Table
List Label	Sample Apex List without Parent	List Icon	standard:lightning_component
Maximum Number of Records	2,000	Number of Records per Page	10
Selectable Rows	<input checked="" type="checkbox"/>	Max Row Selected	2,000
Display Filters Panel All the Time	<input type="checkbox"/>	Show SOSL Search	Not Displayed
		Default Sort Direction	asc

► Child Lists - Relationship Settings

▼ Table Settings

Show Table Header	<input checked="" type="checkbox"/>	Wrap Text Max Lines	
Show Row Number Column	<input type="checkbox"/>	Row Number Start	0

► Tiles Settings

▼ SOQL with/without Sharing & Apex Settings

SObject			
Enable all Record Actions	<input type="checkbox"/>	Enable New Record Action	<input type="checkbox"/>
Enable Edit Record Action	<input type="checkbox"/>	Enable Delete Record Action	<input type="checkbox"/>
Default Visibility Filter	All	Visibility Filter - All	<input checked="" type="checkbox"/>
Visibility Filter - My	<input checked="" type="checkbox"/>	Visibility Filter - My Team	<input checked="" type="checkbox"/>
Visibility Filter - My Subordinates	<input type="checkbox"/>		

► Files Settings

▼ Apex Settings

Data Provider Class	ApexDataSource	Row Key	RowKey
---------------------	----------------	---------	--------

Notes:

- Data Provider Class is the name of the class created at Step 1
- RowKey is the name of the field populated in the Apex Data Source. It must contain a unique row identifier

Step 3: Create the fields

Field				List Settings			Filters Settings		
Field Label	Field Name	Display Type	Hyperlink to Detail Id Field	Display in List	Display Position in List	Sortable in List	Default Sort Field	Display in Filters Panel	Display Position in Filters Panel
Account	Account.Name	Hyperlink to Detail	Id	TRUE	0	Yes	TRUE	Yes	0
Stage	StageName	String		TRUE	1	Yes	FALSE	Yes	1
Number of Opportunities	Count(Id)	Integer		TRUE	2	Yes	FALSE	Yes	2
Total Amount	Sum(Amount)	Currency		TRUE	3	Yes	FALSE	Yes	3
Owner	Owner	String		TRUE	4	Yes	FALSE	Yes	4

Note:

- Field Label and Display Type are required for an Apex Data Source
- Hyperlink to Detail Id Field is required for Apex Data Sources if Display Type = 'Hyperlink to Detail'. It must be populated with the name of the data source field containing the Id of the target record

Step 4: Create the predefined filters

Label	Filter Label	Default Filter	SOQL Filter
Apex - Won	Won Opportunities	FALSE	IsClosed = true AND isWon = true
Apex - Working	Opportunities in Progress	TRUE	IsClosed = false
Apex - Lost	Lost Opportunities	FALSE	IsClosed = true AND isWon = false

Step 5: Add the list to your Home Page or to a custom tab

Test Class

```
@isTest
public with sharing class ApexDataSourceTest {
    @isTest
    static void testGetPage() {
        // Create the test data
        Account acc = new Account(name = 'test');
        insert acc;
        Opportunity oppty1 = new Opportunity(name = 'Oppty1', AccountId = acc.Id, CloseDate = System.today(), StageName = 'Prospecting', Amount = 100);
        Opportunity oppty2 = new Opportunity(name = 'Oppty2', AccountId = acc.Id, CloseDate = System.today(), StageName = 'Value Proposition', Amount = 100);
        Opportunity oppty3 = new Opportunity(name = 'Oppty3', AccountId = acc.Id, CloseDate = System.today(), StageName = 'Closed Won', Amount = 100);
        List<Opportunity> optties = new List<Opportunity>{oppty1, oppty2, oppty3};
        insert optties;
        // Create values from Filters Panel
        smartLists.SmartListController.FilterEntry fe1 = new smartLists.SmartListController.FilterEntry();
        fe1.fieldName = 'StageName';
        fe1.operator='LIKE';
        fe1.values = new String[]{'Prospecting' };
        fe1.type='STRING';
        smartLists.SmartListController.FilterEntry fe2 = new smartLists.SmartListController.FilterEntry();
        fe2.fieldName = 'Sum(Amount)';
        fe2.operator='>=';
        fe2.values = new String[]{'100'};
        fe2.type='CURRENCY';
        smartLists.SmartListController.FilterEntry fe3 = new smartLists.SmartListController.FilterEntry();
        fe3.fieldName = 'Count(Id)';
        fe3.operator='>=';
        fe3.values = new String[]{'1'};
        fe3.type='INTEGER';
        List<smartLists.SmartListController.FilterEntry> fes = new List<smartLists.SmartListController.FilterEntry>{fe1, fe2, fe3};
        Test.startTest();
        // Define getPage parameters
        smartLists.SmartListApexSourceGetPage getPageParms =
            new smartLists.SmartListApexSourceGetPage().withScope('everything').withFilter('').withFilterEntries(fes).withSortField('StageName').withSortDirection('asc').withOffset(0).withPageSize(10);
        ApexDataSource ds = new ApexDataSource();
        // Invoke getPage
        List<Object> result = ds.getPage(getPageParms);
        Test.stopTest();
        System.assertEquals(1, result.size());
    }

    @isTest
    static void testGetRecord(){
        // Create the test data
        Account acc1 = new Account(name = 'test1');
        Account acc2 = new Account(name = 'test2');
```



```
List<Account> accs = new List<Account>{acc1, acc2};
insert accs;
Opportunity oppty1 = new Opportunity(name = 'Oppty1', AccountId = acc1.Id, CloseDate = System.today(), StageName = 'Prospecting');
Opportunity oppty2 = new Opportunity(name = 'Oppty2', AccountId = acc2.Id, CloseDate = System.today(), StageName = 'Value Proposition');
List<Opportunity> oppties = new List<Opportunity>{oppty1, oppty2};
insert oppties;
Test.startTest();
ApexDataSource ds = new ApexDataSource();
// Invoke getRecord
List<Object> result = ds.getRecord(acc1.Id);
Test.stopTest();
System.assertEquals(1, result.size());
}
}
```

Pre-requisites for adding standard record actions to your list

List Definition

SObject: must be populated with the base object

Row Key: can be the Id field for non-aggregate lists (see Field Definitions below)

Field Definitions

You must create a field definition for the record id: Field Name = Id

For objects with record types, you must create a field definition for the record type id: Field Name = RecordTypeId

Apex Data Source

The data source must return the record id in the Id field

For objects with record types, the data source must return the record type id in the RecordTypeId field

Interface and Classes Reference

smartLists.SmartListApexSourceInterface2 Interface

Interface that must be implemented by the data source

The data source class must be defined as global to be visible by the package

Name	Purpose	Method Invoked when	Description	Parameter Type
getPage	Returns one page of records	list is displayed for the first time, is refreshed or scroll is used for displaying more records	Parameters of the page	smartLists.SmartListApexSourceGetPage
getRecord	Returns one record	Record must be refreshed after execution of a row action You don't need to add code to this function if your list does not use row actions	Record Id	String

smartLists.SmartListApexSourceGetPage Class

Parameters of a call to smartLists.SmartListApexSourceInterface2.getPage.

This class includes a fluent builder for creating new instances of parameters for the test classes. Example: new smartLists.SmartListApexSourceGetPage().withScope('everything').withFilter('Status = \'Closed\').withXYZ...

Property	Description	Type	Read	Create
scope	Visibility filter of the query: <ul style="list-style-type: none">- everything: All records- my: My records- team: My team records- subordinates: My subordinates records Notes: <ul style="list-style-type: none">- everything, my, team are SOQL scopes; must be added to USING SCOPE. See: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_select_using_scope.htm- subordinates is not a SOQL scope; must be added to the WHERE clause as Owner.UserRole.ParentRoleId = \' + UserInfo.getUserRoleId() + \'	String	getScope()	withScope(scope)
filter	Predefined filter selected in the list; must be added to the WHERE clause Null if no filters were added to the list	String	getFilter()	withFilter(filter)
parentIdField	Name of the field of the list object containing the Id of the parent record; if populated must be added to the WHERE clause with parentId Only populated for child lists; null otherwise	String	getParentIdField()	withParentIdField(parentIdField)
parentId	Id of the parent record; if populated must be added to the WHERE clause with parentIdField Only populated for child lists; null otherwise	String	getParentId()	withParentId(parentId)
filterEntries	List of filters entered in the Filters Panel and the SOSL search box; must be added to the WHERE clause smartLists.SmartListController.buildFilter(filterEntries) is used for converting the filter entries into a string Null if no filters added	List<smartLists.SmartListController.FilterEntry>	getFilterEntries()	withFilterEntries(filters)

sortField	Sort field selected in the list; must be added to the SORT clause if populated Null if no sort field selected	String	getSortField()	withSortField(sortField)
sortDirection	Sort direction selected in the list: asc or desc; must be added to the SORT clause if populated	String	getSortDirection()	withSortDirection(sortDirection)
offset	Row number of the first record returned by the query; must be added to the query for paging	Integer	getOffset()	withOffset(offset)
pageSize	Number of records of the page; must be added to the query for paging	Integer	getPageSize()	withPageSize(pageSize)

smartLists.SmartListController.FilterEntry Class

Filter entered in the Filters Panel or the SOSL Search box

Property	Description
fieldName	Name of the field in the data source or 'SOSLSearch' for the value entered in the SOSL Search box
operator	Determined by the component from the Display Type: <ul style="list-style-type: none"> - BOOLEAN: = - STRING, EMAIL, PHONE, TEXTAREA, URL: LIKE - DATE, DATETIME, TIME, CURRENCY, DECIMAL, PERCENT, INTEGER: >= OR <=
values	Array of 1 value
Type	Field Display Type

Localization

Overview

The data displayed in the lists is automatically localized. Picklist and Record Types values are translated in the user language if a translation is provided in the Translation Workbench

Date and number fields are displayed according to the user locale.

The labels of the list, fields, filters, and actions are translatable. This can be done by entering \$Label.MyCustomLabel in the corresponding Label field.

See [Step 1: Create a Smart List Definition](#) for example.

All the strings and messages used by the component are translatable custom labels

Component	Description	Label	Value
Global	Assistive text for spinners	Loading	Loading...
Global	Assistive text of dialogs close button	Close	Close
Global	Cancel button label of dialogs and Filters Panel	Cancel	Cancel
SOSL Search box	Placeholder	SearchBoxPlaceholder	Search this list...
SOSL Search box	Assistive text	SearchBoxLabel	Search when user hits the 'enter' key
Custom Filters combobox	Assistive text of the combobox	FilterSelection	Filter Selection
Visibility Filters combobox	Assistive text of the combobox	VisibilityFilterSelection	Visibility Filter Selection
Visibility Filters combobox	All filter label	SOQLScopeAll	All
Visibility Filters combobox	My filter label	SOQLScopeMy	My
Visibility Filters combobox	My Team filter label	SOQLScopeMyTeam	My Team
Visibility Filters combobox	My Queues filter label	SOQLScopeMyQueues	My Queues
Visibility Filters combobox	My Subordinates filter label	SOQLScopeMySubordinates	My Subordinates
Tile – Sort Field Selector	Assistive text of the Sort Field Selector	SortFieldSelection	Sort Field Selection
Tile – Sort Field Order	Assistive text of the Sort Ascending button	SortAscending	Sort Ascending
Tile – Sort Field Order	Assistive text of the Sort Descending button	SortDescending	Sort Descending
Refresh list button	Button title	Refresh	Refresh
Show/Hide Filters button	Button title of selected state	HideQuickFilters	Hide Filters
Show/Hide Filters button	Button title of deselected state	ShowQuickFilters	Show Filters
List Info	Label for several items	ItemPlural	Items
List Info	Label for one item	ItemSingular	item
List Info	Label for several items selected in the list	SelectedPlural	selected
Filters Panel	Label for several items selected in the picklist values dialog		
List Info	Label for one item selected in the list	SelectedSingular	selected

Filters Panel	Label for one item selected in the picklist values dialog		
List Info	Label for sort field	SortedBy	Sorted by
List Info	Label of filter by criteria	FilteredBy	Filtered by
List Message	Message displayed at the bottom of the list when more records are available	LoadMoreRecords	Scroll to load more records
List Message	Message displayed at the bottom of the list when the maximum number of records has been loaded	MaxRecordsLoaded	Maximum of {0} records loaded. Additional records may be available
List Message	Message displayed at the bottom of the list when no records found	NoItemsToDisplay	No items to display.
List Message	Message displayed at the bottom of the list when all records have been loaded	AllRecordsLoaded	All records are loaded
Filters Panel	Filter widget title	QuickFilters	Filters
Filters Panel	SOSL search field label - Files	FilterSOSLSearchFileContent	Search File Content
Filters Panel	SOSL search field label - Record	FilterSOSLSearchRecord	Search Record
Filters Panel	Date/Datetime/Time range: Start filter label	FilterRangeStartLabel	Start
Filters Panel	Date/Datetime/Time range: End filter label	FilterRangeEndLabel	End
Filters Panel	Number Range: Min filter label	FilterRangeMinLabel	Min
Filters Panel	Number Range: Max filter label	FilterRangeMaxLabel	Max
Filters Panel	Boolean filter: checked value label	Checked	Checked
Filters Panel	Boolean filter: unchecked value label	Unchecked	Unchecked
Filters Panel	Picklist filter: label of the Show More button displayed when the picklist has more than 5 values	ShowMore	Show More
Filters Panel	Lookup filter: search box placeholder when no data has been entered	LookupPlaceholder	Search {0}...
Filters Panel	Lookup filter – Search results: recent records label	LookupRecent	Recent {0}
Filters Panel	Lookup filter – Object selector: accessible text	LookupObjectSelector	Select an object - Current Selection: {0}
Filters Panel	Lookup filter – Object selector: queue object label	LookupQueueus	Queues
Filters Panel	Lookup filter – Search results: no records found message	LookupNoResults	No Results
Filters Panel /Field label in List	Label of the Owner field	Owner	Owner
Filters Panel	Clear filter button assistive text	Clear	Clear
Filters Panel	Apply button label	Apply	Apply
Filters Panel	Clear All Filters button label	ClearAllFilters	Clear All Filters
Filters Panel / SOSL Search box	SOSL search field: Error message for value with less than 2 characters	FilterSOSLSearchTooShortError	The search string must be at least 2 characters
Filters Panel	Date/Datetime/Time range: Error message for start value > end value	FilterRangeStartError	Start value must be lower than End value
Filters Panel	Date/datetime range: Error message for end value < start value	FilterRangeEndError	End value must be greater than Start value
Filters Panel	Number range: Error message for min value > max value	FilterRangeMinError	Min value must be lower than Max value
Filters Panel	Number range: Error message for max value < min value	FilterRangeMaxError	Max value must be greater than Min value

Filters Panel – Select Picklist values dialog	Label of Available picklist values	Available	Available
Filters Panel – Select Picklist values dialog	Label of Selected picklist values	Selections	Selections
New record list action	New button label	New	New
Edit Record dialog - New	Next button label on the record type page	Next	Next
Edit Record dialog – New	Message displayed on save success	was created.	was created.
Edit Record dialog Edit File Details dialog Edit record row action	Dialog title Dialog title Edit menu item label	Edit	Edit
Edit Record dialog - Edit Edit File Details dialog	Save button label	Save	Save
Edit Record dialog	Label of required field description	RequiredInformation	Required Information
Edit Record dialog	Message displayed if page level errors after save	ReviewPageErrors	Review the errors on this page
Edit Record dialog	Message displayed if field level errors after save	ReviewFieldsErrors	Review the following fields
Edit Record dialog	Assistive text for save in progress	Saving	Saving...
Edit Record dialog Edit File Details dialog	Message displayed on save success	WasSaved	was saved.
Delete record row action Delete file row action	Menu item label	Delete	Delete
Delete Record dialog	Dialog title	DeleteRecordTitle	Delete {0}?
Delete Record dialog	Confirmation message for record deletion	DeleteRecordMessage	Are you sure you want to delete this {0}?
Delete Record dialog	Confirmation message for file deletion	DeleteFileMessage	Deleting a file also removes it from any records or posts it's attached to
Delete Record dialog	Message displayed on successful record delete	WasDeleted	was deleted.
Upload Files list action	Button label	UploadFiles	Upload Files
Upload Files list action	Message displayed on successful file upload	FilesHasBeenUploaded	File(s) have been uploaded
Download list action	Menu assistive text	DownloadMenu	Download Menu
Download Files dialog	Confirmation message	DownloadConfirm	Are you sure you want to download {0} file(s) worth {1} MB?
Download Files dialog	Dialog title	DownloadFiles	Download Files
Download Files dialog	Error message when download files limit exceeded	DownloadLimit	Download limit exceeded. Please download less than {0}MB and less than {1} files
Download All Files list action	Menu item label	DownloadAllFiles	Download All Files
Download Selected Files list action	Menu item label	DownloadSelectedFiles	Download Selected Files
Preview Selected Files list action	Button assistive text	PreviewSelectedFiles	Preview Selected Files
Download File row action	Menu item label	Download	Download
Edit Files Details row action	Menu item label	EditFileDetails	Edit File Details
Preview file row action	Menu item label	PreviewFile	Preview File
Upload New Version row action	Menu item label	UploadNewVersion	Upload New Version
Upload New Version row action	Message displayed after a successful upload of a new version	FileVersionWasUploaded	File "{0}" was uploaded

Upload New Version dialog	Label of Upload button	Upload	Upload
Upload New Version dialog	Label of new version reason field	UploadNewVersionReason	What Changed? (optional)
Upload New Version dialog	Error message for technical error on upload new version	ErrorMsgCantUploadNewVersion	Unexpected error on upload new version:
Upload New Version dialog	Error message for invalid file extension	ErrorMsgInvalidExtension	Your company doesn't support the following file types:
View Files Details row action	Menu item label	ViewFileDetails	View File Details
Smart Files List dialogs	Used in dialogs and messages	File	File
Smart Files List	Default label of files list	Files	Files
Tiles	Load More button label	LoadMore	Load More
Tiles	Load All button label	LoadAll	Load All
Row selection – Screenflow context	Error message on selected records < min selected records	MinRowSelectionError	You must select at least {0} record(s)
Inline Edit	Inline Edit Errors	Errors	Errors(s)
Inline Edit	Inline Edit changes were saved message	SavedChanges	Your changes has been saved
Inline Edit	None value for picklist	NoneValue	--None--