Search documentation

Appian

Record Design

Appian 7.6

The Records tab brings together all data used within Appian into a collection of information through the form of lists and record dashboards.

Toggle Us

This includes both integrated business data, such as customer information or payroll information, and process data, such as Support Tickets created through an Action.

The sections below detail design information for records, such as what to expect in terms of record security, the options available when configuring a record type, and how to manage records after their creation.

Tu

Re

Ot

To walk through an example of creating your first record, see also: Records Tutorial

Frequently Asked Questions

What are records?

Records bring Appian into a data-centric design model and extend support for Dynamic Case Management and Business Process Management soluti Through the use of rules, SAIL components and expressions, records enable designers to quickly define a common and secure method for users to navigate, collaborate, report upon, and take action upon enterprise data.

How do records simplify traditional process modeling?

We expect many data-centric use cases based on records to simplify the design and performance, and testing of process models. As an example, many formerly "long-running processes" can be shortened by maintaining data in a record instead of in the process instance.

What happens to my existing application designs?

These continue to work as before in the legacy Portal interface as part of Appian 7.x. As you migrate your application to Tempo, you can design record dashboards to replace each process dashboard and access them from the Tempo interface.

Are records a substitute for process dashboards?

Yes. Records are a new way of conveying process information, as well as other business data, to users using Appian's new SAIL framework. This new framework gives designers control over how information is presented through the configuration of record list views and record dashboards available onl to records. These new user interfaces (discussed later) are also viewable on mobile devices and the broad set of web browsers supported by Tempo.

Can users search for records?

By default, users can perform a record name keyword search on records from the record list view for a record type. The search is based on the data used for the record name in the record list view.

For service-backed records, you can enable search by configuring it in the record source expression.

What is the difference between a CDT and a record?

A CDT is a data definition for developers who are integrating with external data; records are an end-user facing feature which exposes a data-centric view. If a record is based on a data store entity, then the underlying data is stored in CDTs.

Can I import/export records?

Records are like other Appian objects in that their definitions can be deployed between development and production systems using application containers.

How do I connect records to my enterprise data sources?

You can create records from enterprise data sources using service-backed records as discussed later in the Source section.

Do I have to use records when I upgrade?

No, you can update to the latest version of Appian without any required re-work. You can begin using records when you choose.

Record Concepts

Record Type

A record type is an Appian object containing metadata for a given record including the source, record list view, record dashboards, default filters, facets and related actions common to all records of the same source.

Record

A record is an instance of a record type. For example, a record type called Prospectives could include a record for each prospective customer in the system. A record type called Help Desk Tickets could include a record for each ticket created through a technical support action.

Source

The source is what supplies the data for the record. There are three types of records, each with a different type of source:

Entity-Backed Records

An entity-backed record is sourced from a data store entity and generates a record for each row in the top-level data store entity.

Process-Backed Records

A process-backed record is sourced from a process model and generates a record for all processes for that process model.

Service-Backed Records

A service-backed record is sourced from an expression and generates records based on the output of that expression. For example, data from a web service can be turned into records by creating a service-backed record and using a custom plug-in function in the source expression to retrieve the dat from the web service. Default filters and facets for service-backed records are also defined via expression, which is more flexible and dynamic than the default filter and facet definitions for entity-backed and process-backed records. When you need more flexibility with your default filter and/or facet definitions and/or if your facets and facet options are likely to change over time, consider choosing to build your record as a service-backed record rath than an entity-backed record. Also note that `queryrecord()` cannot reference service-backed records, so use entity-backed records when this function in use.

Record List View

The record list view displays all of the records of a given record type that a user has the right to view and any facets configured for the record type. Users access the records from this list.

Only the first 100 records display in the record list view.

Facets

Facets are filters end users can apply to the record list view to view a subset of the records. For example, on a customer record type, you might create facet called `Status` where the facet options are `Prospective`, `Active`, and `Inactive` by using the `=` operator.

Record Dashboards

A record dashboard is an interface for users to view data for a specific record. You can create multiple dashboards for a single record type. Users access them by selecting a record from the record list view. The SAIL summary dashboard displays by default. Users access all other SAIL dashboards configured for the record from the links on the left.

The layout and data that display for each record is determined by the expressions used for the SAIL Summary Dashboard and SAIL Dashboards fields configured as part of the record type.

Related Actions

Related actions allow users to act on a record or in the context of that record. Process-backed records derive related actions from the model's quick tasks, and records of any source type can be configured to have related actions that start process models. For example, for a record that represents a customer, there might be a related action to enter a new order for that customer or to update the information about that customer.

Security Model

Security for a record can be defined at various levels. This includes access to its record type in Tempo, accessing the record, viewing specific informatio on the record, viewing the other record dashboards, and starting the record's related actions.

Record Type Security

Each record type has a rolemap specifying its Viewers, Auditors, Editors, and Administrators. Only one group can be added to each role. Users must als have at least viewer rights to the source to view a record in Tempo. By default, only the record type creator has access to the record type.

Users in each role are allowed to perform the following actions on the record type:

Actions / Roles	Administrator	Editor	Auditor	Viewer
View in Tempo	Yes	Yes	Yes	Yes
View properties	Yes	Yes	Yes	Yes
View security	Yes	Yes	Yes	No
Update properties	Yes	Yes	No	No
Update security	Yes	No	No	No
Delete object	Yes	No	No	No

To access the record type's definition and rolemap, users must also be a System Administrator.

Notes

If users have viewer rights to the record's source, restricting them from viewing records through record type security does not stop them from viewin the associated data in other areas of the system.

When importing a record type into an environment where the record type already exists (e.g. to update it), the imported record type's rolemaps will b merged with the existing record type's rolemaps. If the imported record type has different rolemaps from the existing record type, you will need to reconfigure the security on the environment after the import using the record type designer interface. The record type designer interface only displays one group per role, so it may not accurately reflect the rolemaps after import if the import resulted in multiple groups for a single role. Reconfiguring the security for all of the roles will ensure that each role has the correct rolemap set and displayed. This behavior may change in a future release.

See also: Managing User Rights and Security

Record Security

Record security is based on the security of the underlying source. Users must have at least viewer rights to the record's source to view the record in th record list view or to view its record dashboards.

For example, applying a default filter to hide a specific group of processes will stop those records from generating, but the process data might still be visible in a Portal report or process dashboard.

The security of the record's source is configured differently for the different source types:

For entity-backed records, see also: Configuring Data Store Security

For process-backed records, see also: Configuring Process Security

For service-backed records, the source expression executes in the context of the user viewing the record list view. Even if the source expression is defined using a rule, the security role map applied to that rule does not prevent any users from executing the rule by viewing the record list view. Acces to the underlying data must be controlled by the designer of the rule in its definition in conjunction with the access control mechanisms available from t provider of the data. For instance, if the rule retrieves data from an external data provider that requires credentials for authentication and authorization, the rule designer must build the retrieval and presentation of those credentials into the definition of the rule.

See also: Expression Rule Security

Record Dashboard Security

Security for record dashboards is a combination of the source security, record type security, and filter configurations.

A user must be able to view a record in the record list view to to access the record. Any one with access to the record will see the SAIL summary dashboard by default. If a user does not have access because of source security, record type security, or a filter configuration, the user cannot access the record dashboards even if given a direct URL.

Each additional SAIL dashboard also has its own security. This is based on the Visibility Expression defined for the dashboard.

NOTE: Hiding data on a dashboard does not secure the underlying data. It only determines what does not display on the dashboard.

Related Actions Security

Security for starting related actions is based on the security of the underlying process model. Users can only start a related action if they can view the record and have initiator rights to the action's process model. The same applies for quick tasks that appear as related actions for process-backed recorc If the user does not have the rights to complete the quick task, the link to the related action will not display under Related Actions.

See also: Configuring Process Security

Creating a Record Type

Creating a record type involves four separate steps:

1. **Configure the Record List View**. An expression that maps record field data into a List View item for display in the record list view.
2. **Configure the Security Groups**. Similar to any other group in Appian and allow you to easily change the security for the record type.
3. **Configure the SAIL Summary Dashboard**. Interface rule for a SAIL interface that determines the layout and components to display on the summary dashboard for each record. You can created additional record dashboards at any time.
4. **Create the Record Type Object**. The interface for this is located on the Data Management page of the Designer interface. Once completed, tł record type and its records display in Tempo for users with at least viewing rights.

Configure the Record List View

To determine what information displays for each record when users view the record list view, configure a ListViewItem as detailed below.

See also Constructing Data Type Values

Syntax

```
=a!listViewItem(
  image: <value>,
  title: <value>,
  details: <value>,
  timestamp: <value>
)
```

Fields

- *image* (Document or User): Optional image associated with the record. If left null or empty, the first two letters of the record title display. For document values, a thumbnail of the document displays. For user values, the user's avatar displays.
- *title* (Text): Title of the record.
- *details* (Text): Optional description of the record.
- *timestamp* (Date and Time): Optional timestamp for the record, such as creation timestamp or last modified timestamp.

Notes

Appian recommends saving this expression as a rule for version control and testing purposes. This rule will be used as the List View Item Template value for your record type.

The fields of the record are available to the list view item template expression under the `rf` (or "record field") expression context.

When users search for records, only the `rf` fields of type Text and Integer that are used in the *title* value are searched. Avoid using fields of any other type in the *title* field because users may try to search on their values, but the expected records won't return in the search results.

Record fields of complex types have their nested fields available through standard dot notation.

For example, the zipCode field of a data store entity of type Person is accessible like this:

```
rf!address.zipCode
```

However, if a field contains multiple complex values, you cannot access a field inside the list using dot notation.

For example, if you have a field named employees of type list of People, you cannot access a list of the addresses for each employee using dot notation `rf!employees.address` will result in an error. The recommended way to acquire the list of addresses is to store the employees field as a local variable usinç the `with()` function.

See also: with()

For entity-backed records, the fields of the underlying data type of the entity are available as fields of the record.

For service-backed records, the fields of the corresponding source data type are available as fields of the record.

For process-backed records, all non-hidden process variables are available as fields of the record, except those from a sub-process. Referencing a sub-process variable will result in the same error as accessing a process variable that does not exist.

The field name is the process variable name. Any referenced process variable must exist in the latest published version of the process model. If a proces variable exists in the latest version of the model but not in an older version, references to the new process variable in the old version (`rf!newVarName`) ' be blank.

In addition to process variables, process-backed records also have process properties and process model properties exposed as record fields named `pp` and `pm` , respectively.

The tables below illustrate how to access process and process model properties in the `rf!` domain.

Process Property	Record Field	Record Field Type

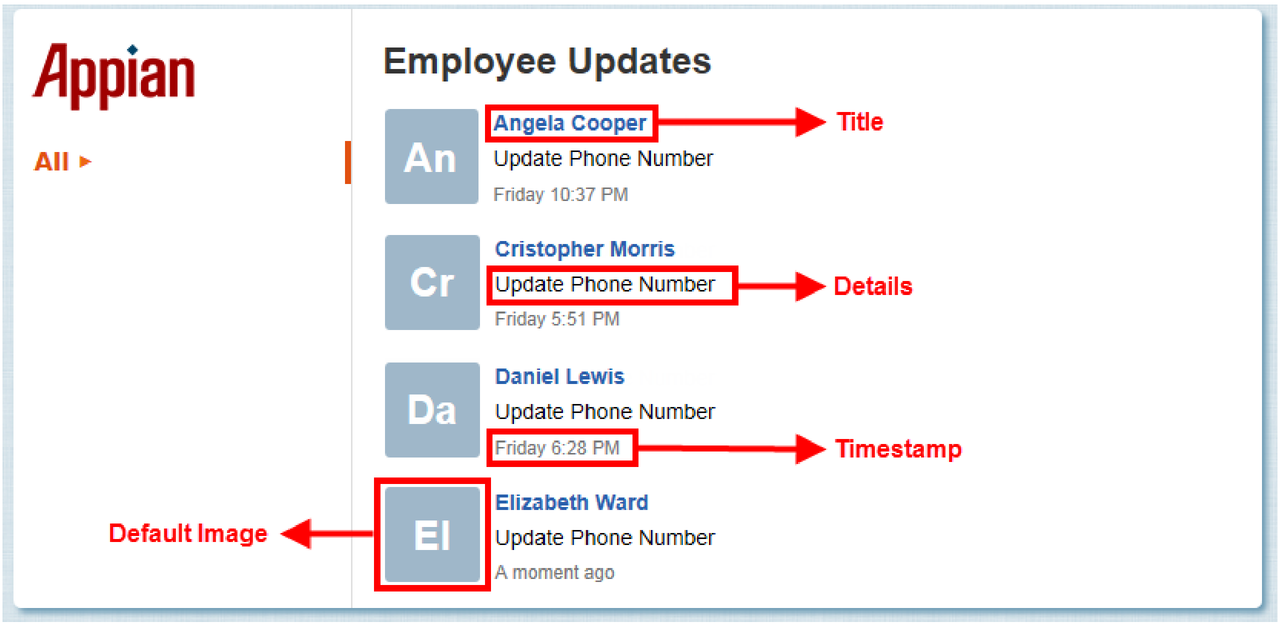
pp!id	rf!pp.id	Any Type
pp!name	rf!pp.name	Any Type
pp!priority	rf!pp.priority	Any Type
pp!initiator	rf!pp.initiator	Any Type
pp!starttime	rf!pp.startTime	Any Type
pp!deadline	rf!pp.deadline	Any Type
pp!timezone	rf!pp.timeZone	Any Type
pp!description	rf!pp.description	Any Type
is_process_favorite()	rf!pp.starred	Any Type
parent_process_id()	rf!pp.parentId	Any Type
parent_process_name()	rf!pp.parentName	Any Type
process_ee_id()	rf!pp.execId	Any Type
process_status()	rf!pp.status	Any Type
process_completion_time()	rf!pp.endTime	Any Type

Process Model Property	Record Field	Record Field Type
pm!id	rf!pm.id	Any Type
pm!uuid	rf!pm.uuid	Any Type
pm!name	rf!pm.name	Any Type
pm!description	rf!pm.description	Any Type
pm!version	rf!pm.version	Any Type
pm!creator	rf!pm.creator	Any Type
pm!timezone	rf!pm.timeZone	Any Type
is_pm_favorite()	rf!pm.starred	Any Type
parent_pm_id()	rf!pm.parentId	Any Type
parent_pm_name()	rf!pm.parentName	Any Type

The types of these record fields might change in a future release.

Examples

```
=a!listViewItem(
  title: rf!employeeToUpdate.name,
  details: rf!pm.name,
  timestamp: rf!pp.startTime
)
```



Configure the Security Groups

As discussed in the Security Model section, only one group can be added to each security role of the record type rolemap.

Appian recommends creating groups specifically for the record type.

Create a group for each of the following roles and add users or groups to them as needed:

- Administrators
- Editors
- Auditors
- Viewers

Configure the SAIL Summary Dashboard

To configure your summary dashboard, create an interface rule using SAIL components. To display record data on the dashboard use the record type fields as values for the SAIL component fields.

For details about the `rf` domain, see above: Configure the List View

See also: Creating an Interface and SAIL Tutorial

This expression will be used as the SAIL Summary Dashboard value when creating the record type object.

For example, a SAIL Summary Dashboard value that returns the Name, Status, and Priority of a customer record may resemble the following:

```
=a!dashboardLayout(
  firstColumnContents: {
    a!textField(
      label: "Customer",
      readOnly: true,
      value: rf!name
    ),
    a!textField(
      label: "Status",
      readOnly: true,
      value: rf!status
    ),
    a!textField(
      label: "Priority",
      readOnly: true,
      value: rf!priority
    )
  }
)
```

The same rules apply for accessing fields for the record type using the `rf!` domain as stated in the "Configure the Record List View" section above.

NOTE: Appian recommends saving the expression as a rule and then calling the rule as the SAIL Summary Dashboard value to take advantage of the version control and security features of rules.

See also: SAIL Tutorial

Create the Record Type Object

To create the record type object, access the Record Types tab from the Data Management page of the System view and click Create Record Type.

Fields

- *Name*: The name of the record type for finding it in the Designer interface. Only accepts a Text value. For example, Prospective, Customer, or Support Ticket.
- *Plural Name*: The name of the record type in plural form. Only accepts a Text value. For example, Prospectives, Customers, or Support Tickets. Visible to end users in Tempo under the Records tab.
- *Description* (Optional): The description of the record type shown in the record type list under each record type plural name. Only accepts a Text value. For example, "Prospective customers bringing new business", "Current customers of the business", or "Issues reported by customers".
- *URL Stub* (Optional): The portion of the URL that uniquely identifies this record type. Only accepts a text value. For example, "prospectives," "customers," or "tickets".
- *Source Type*: The source type of the record data. For a service-backed record, select Expression.
- *Source*: The source data for the record. Either a process model, a data store entity, or an expression returning a value of type DataSubset.
- *Source Data Type*: The custom data type corresponding to the record. For service-backed records only. Fields of this data type are available for use with the "record field" (`rf!`) domain.
- *List View Item Template*: Rule you created earlier that determines how the Record List View will display the record types. Include rule inputs as needed.
- *Sort Field*: Field the records are sorted by in the record list view. For service-backed records, the designer controls the sort of the record list view based on what they configure for the record type source.
- *Sort Order*: Direction in which the records are sorted. Options are Ascending or Descending. For service-backed records, the designer controls tl sort of the record list view based on what they configure for the record type source.
- *Viewers* (Optional): Select "All Users" to allow all authenticated users of the system to view the record type or select "Restricted" and specify a single group whose members have viewer rights to the record type.
- *Auditor Group* (Optional): A single group whose members have auditor rights to the record type.
- *Editor Group* (Optional): A single group whose members have editor rights to the record type.
- *Administrator Group* (Optional): A single group whose members have administrator rights to the record type.
- *SAIL Summary Dashboard*: An expression that calls the rule you created for the summary dashboard. Include rule inputs as needed. See above: Configure the SAIL Summary Dashboard.

Notes

The URL stub should be unique, short, and not contain spaces. Supported characters include letters, numbers, dash (-), and underscore (_). If a URL Stub is not provided, the system will automatically generate a random value when you save the record type. For example, "lstM7Q".

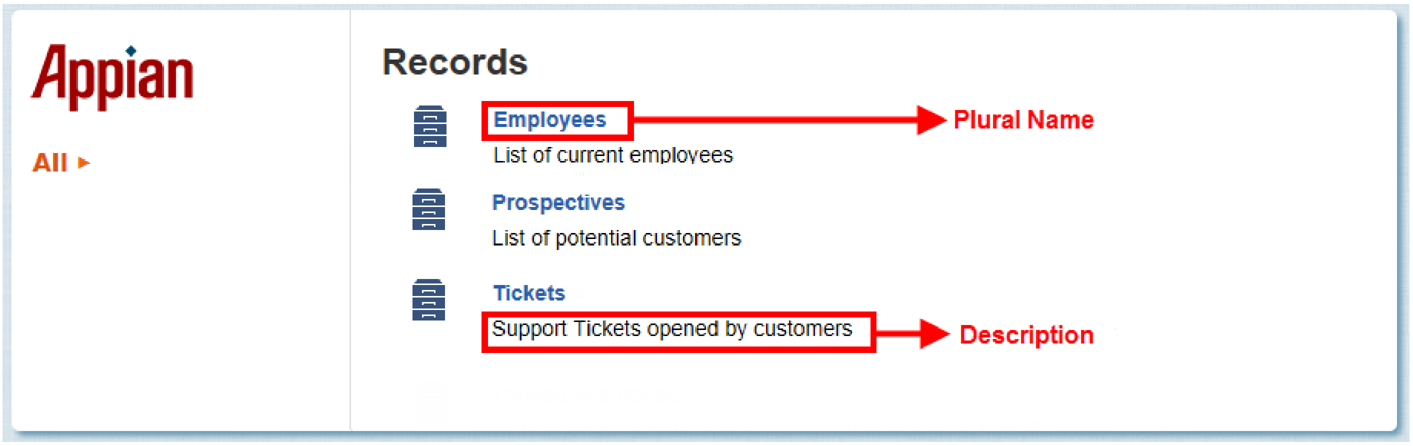
For entity-backed records, the entity you choose as the source must have a defined primary key (as opposed to an automatically generated one).

See also: Indexes and Primary Keys

Once the record type is saved, the records are available on Tempo to all users with at least viewing rights.

Examples

Record Types List with Three Record Types


Appian
All ▸
Records
Employees
List of current employees
Plural Name
Prospectives
List of potential customers
Tickets
Support Tickets opened by customers
Description

Optional Configurations

After creating your record type, you can add the following:

- **Additional Dashboards**: Records dashboards users can access in addition to the summary dashboard that displays when they open a record.
- **Default Filters**: Filters applied to the source data to control which records appear in the record list view. These do not limit access to the underlying data sources.
- **Facets**: Filters an end user can apply to the record list view to restrict which records they see.
- **Related Actions**: Actions an end user can start from the record to act on the record data or with the record as its context.

Additional Dashboards

You can add up to eight additional dashboards for each record type to complement the summary dashboard. Each additional dashboard has its own visibility setting to allow only certain users to access it.

Visibility settings do not provide protection to the underlying data.

See above: Record Level Security

To add an additional dashboard to a record type, access the Record Types tab from the Data Management page of the System view, select your recor type, and click **Create a New Dashboard** from the Dashboards box.

Fields

- *Name Expression*: The expression that determines the name to display for the dashboard in the left-side navigation of the record.
- *SAIL Dashboard*: The expression that determines the look and feel of the dashboard.
- *Visibility Expression*: The expression that determines what users can see the dashboard name and access it.

Notes

Once you click **Create Dashboard**, the dashboard is added to the list and saved to the record type. You do not need to click **Update Record Type** save the new dashboard.

You will have access to `rp!id` as well as the record fields under the `rf!` domain to use in the visibility expression so that you can specify different visibility settings for records of the same type based on their attributes.

To add another dashboard, click **Create a New Dashboard** and add new values for the same fields above.

Default Filters

You can apply multiple default filters to a record type. All filters are joined by an AND union. Only records that pass all default filters appear in the record list view or have record dashboards.

Default filters do not provide protection to the underlying data.

See above: Record Level Security

Process-Backed and Entity-Backed Records

To add a default filter to a process-backed or entity-backed record type, access the Record Types tab from the Data Management page of the System view, select your record type, and click **Create a New Default Filter** from the Default Filters box.

Fields

- *Record Field*: The field of the record type to filter by.
- *Operator*: The comparison operator to filter with.
    - Filters using the `BETWEEN` operator should supply a Value that is a two element list.
- *Value*: The expression which returns values to compare against.

Notes

Once you click **Create Default Filter**, the default filter is added to the list and saved to the record type. You do not need to click **Update Record Typ** to save the new default filter.

To add another default filter, click **Create a New Default Filter** and add new values for the same fields above.

Service-Backed Records

To add default filters to a service-backed record type, access the Record Types tab from the Data Management page of the System view, select your record type, and enter the expression using the `a!queryFilter` function in the Default Filters field. Appian recommends defining the default filters expression as a rule for version control and testing purposes and calling that rule in the Default Filters field.

See also: a!queryFilter()

Facets

Facets are filters end users can apply to the record list to view a subset of the records. An end user may select any number of facets configured for the record type. Multiple facets are applied by an AND union.

Facets are grouped by facet options. Each facet must contain at least one facet option, but can contain many more. For example, a facet named Location might contain two options: America and Europe.

Process-Backed and Entity-Backed Records

To add a facet to a process-backed or entity-backed record type, access the Record Types tab from the Data Management page of the System view, select your record type, and click **Create a New Facet** from the Facets box.

Fields

- *Facet Name*: Name displayed to designers in the Facets list when editing the record type in the Designer interface. Only accepts a Text value.
- *Order Index*: Value to determine where the facet displays in the list of facets for the end users. Only accepts an Integer value and cannot be the same as an existing facet for the same record type. The lower the number, the higher it displays in the left-hand navigation of the record list view
- *Label*: Expression that defines the name of the filter group displayed to end users. For example, "Location" displays as Location.
- *Record Field*: The field of the record type to filter by.
- *Initial Option Label*: Expression that defines the name of the first facet option displayed to end users. For example, "North America" displays as North America.
- *Operator*: The comparison operator to filter with. See also: Comparison Operators
- *Value 1*: An expression that returns the value(s) for the `=` and `<>` operators or the lower bound for the `BETWEEN` operator.
- *Value 2*: An expression that returns the upper bound value for the `BETWEEN` operator. This field only appears on the form if you selected `BETWEEN` as the value for the Operator field.

Notes

The IN operator represents equals.

Once you click **Create Facet**, the facet is added to the list and saved to the record type. You do not need to click Update Record Type to save the ne facet.

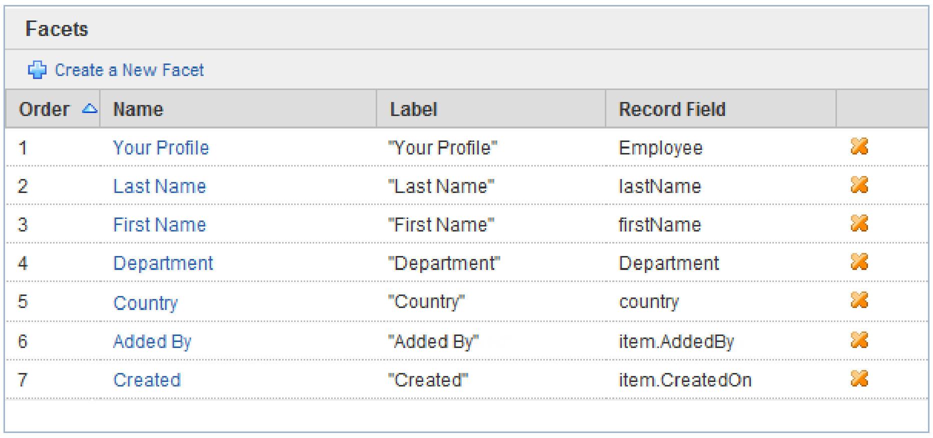
To add another option to the facet, select the facet name and click **Create a New Facet Option.** Enter the facet option values as detailed above and click **Create New Facet Option** followed by **Update Facet.** The options automatically display in the order in which you add them.

To change the order of the facet options, select the facet and click the arrow icons next to a facet option to move it up or down in the list. Clicking **X** deletes the facet option.

To add another facet with a different set of options, click **Create a New Facet** and add new values for the same fields above.

Examples

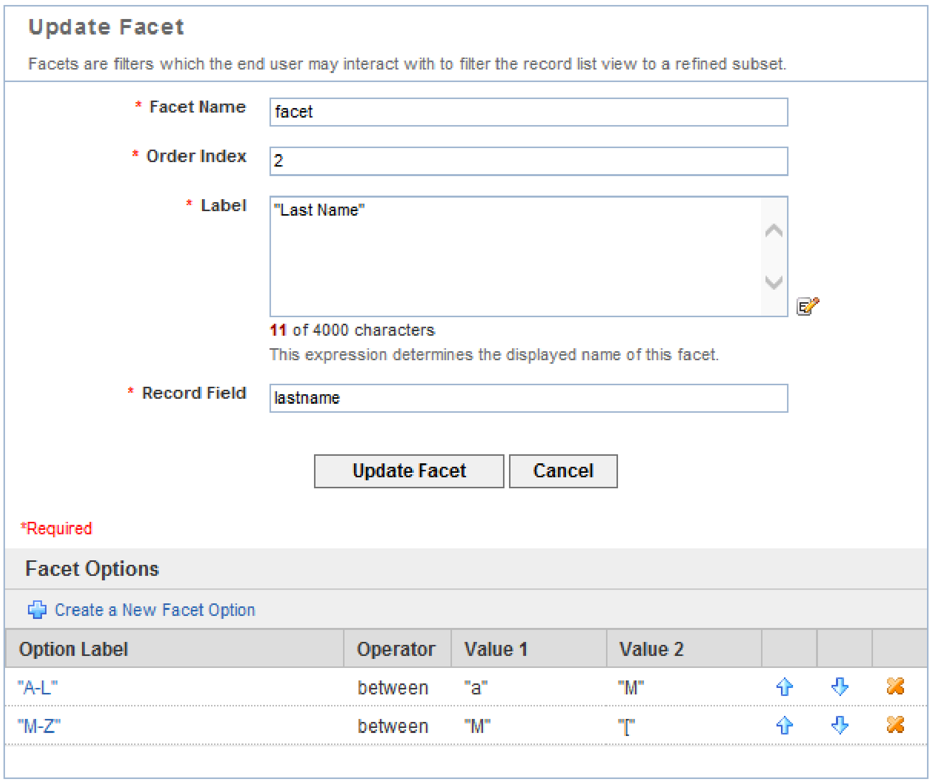
Facet List



Facets

Order	Name	Label	Record Field	
1	Your Profile	"Your Profile"	Employee	✖
2	Last Name	"Last Name"	lastName	✖
3	First Name	"First Name"	firstName	✖
4	Department	"Department"	Department	✖
5	Country	"Country"	country	✖
6	Added By	"Added By"	item.AddedBy	✖
7	Created	"Created"	item.CreatedOn	✖

Facet Option Using BETWEEN Operator with String Comparison



Update Facet

Facets are filters which the end user may interact with to filter the record list view to a refined subset.

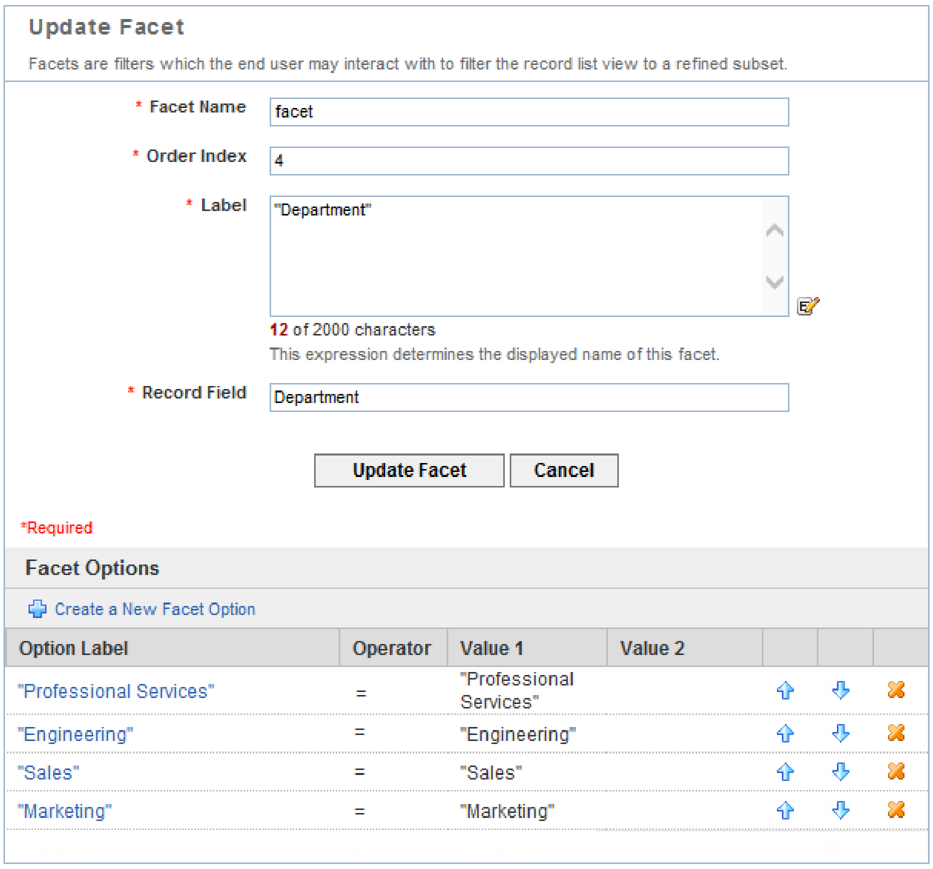
* Facet Name	facet
* Order Index	2
* Label	"Last Name"
	11 of 4000 characters This expression determines the displayed name of this facet.
* Record Field	lastname

*Required

Facet Options

Option Label	Operator	Value 1	Value 2			
"A-L"	between	"a"	"M"	⇧	⇩	✖
"M-Z"	between	"M"	"["	⇧	⇩	✖

Facet Option Using String Comparison with = Operator



Update Facet

Facets are filters which the end user may interact with to filter the record list view to a refined subset.

* Facet Name	facet
* Order Index	4
* Label	"Department"
	12 of 2000 characters This expression determines the displayed name of this facet.
* Record Field	Department

Update Facet | Cancel

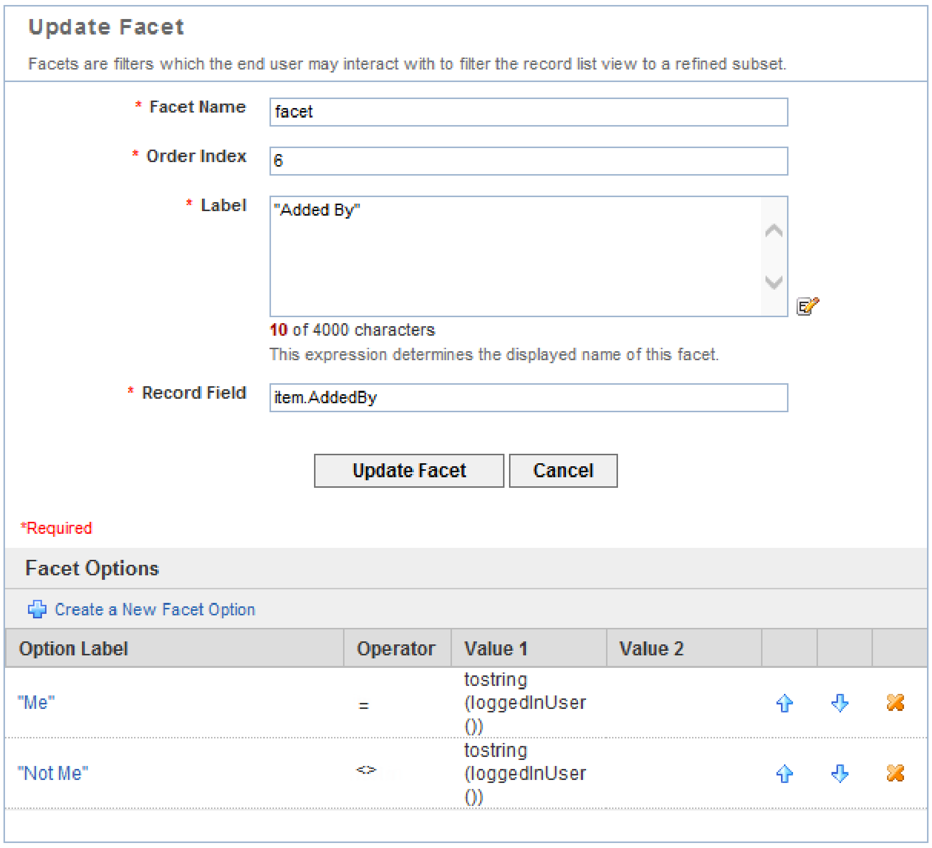
*Required

Facet Options

Create a New Facet Option

Option Label	Operator	Value 1	Value 2			
"Professional Services"	=	"Professional Services"		⇧	⇩	✖
"Engineering"	=	"Engineering"		⇧	⇩	✖
"Sales"	=	"Sales"		⇧	⇩	✖
"Marketing"	=	"Marketing"		⇧	⇩	✖

Facet Option Using User Comparison



Update Facet

Facets are filters which the end user may interact with to filter the record list view to a refined subset.

* Facet Name	facet
* Order Index	6
* Label	"Added By"
	10 of 4000 characters This expression determines the displayed name of this facet.
* Record Field	item.AddedBy

Update Facet | Cancel

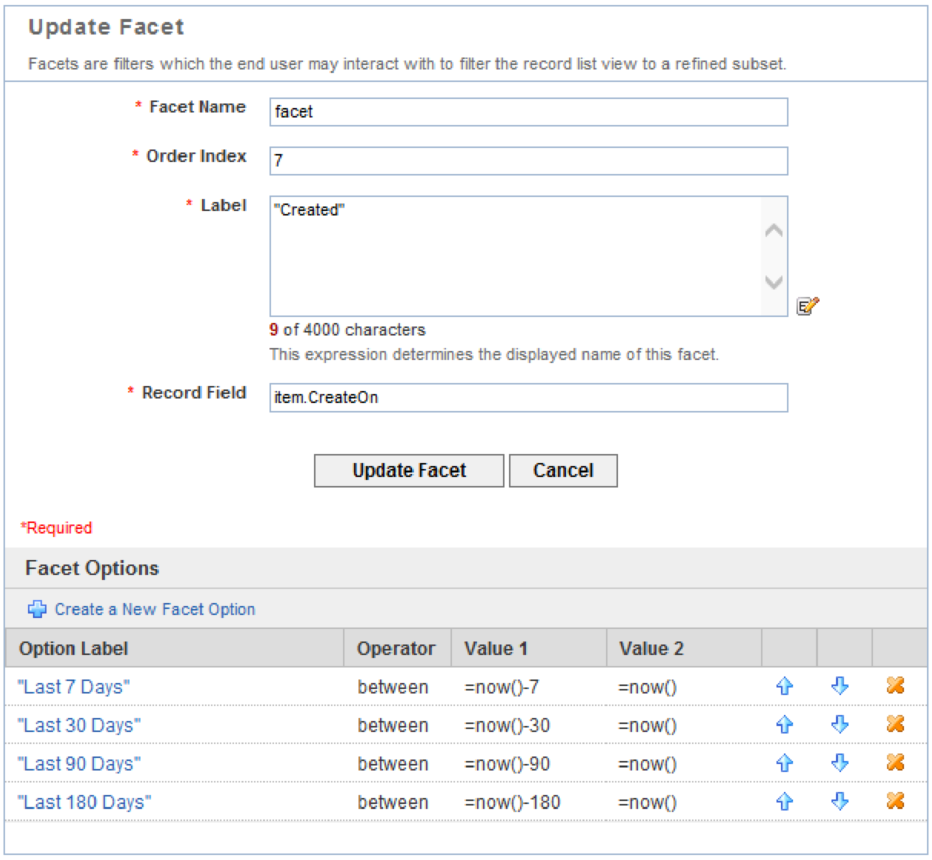
*Required

Facet Options

Create a New Facet Option

Option Label	Operator	Value 1	Value 2			
"Me"	=	tostring (loggedInUser ())		⇧	⇩	✖
"Not Me"	<>	tostring (loggedInUser ())		⇧	⇩	✖

Facet Option Using Data Range Comparison



Update Facet

Facets are filters which the end user may interact with to filter the record list view to a refined subset.

* Facet Name	facet
* Order Index	7
* Label	"Created"
	9 of 4000 characters This expression determines the displayed name of this facet.
* Record Field	item.CreateOn

Facet Options

Option Label	Operator	Value 1	Value 2			
"Last 7 Days"	between	=now()-7	=now()			
"Last 30 Days"	between	=now()-30	=now()			
"Last 90 Days"	between	=now()-90	=now()			
"Last 180 Days"	between	=now()-180	=now()			

Service-Backed Records

To add facets to a service-backed record type, access the Record Types tab from the Data Management page of the System view, select your record type, and enter the expression to define the facets into the Facets field. Appian recommends defining the facets expression as a rule for version control and testing purposes and calling that rule in the Facets field.

Use the `a!facet` function to construct the Facets expression for a service-backed record type. See also: a!facet()

For service-backed records, the framework communicates the user's interactions with the facets through the `query` object passed to the source rule. The filter associated with the selected facet option is passed as a filter in the `logicalExpression|filter|search` field of the `query` object. For service-backed records, you can access the `query` object by using the "record source parameter" (`rsp!`) domain, i.e. `rsp!query` .

See also: Query Data Type and Tutorial: Add a Facet to Service-Backed Records

Related Actions

For each record type, you can define a set of related actions based on process models. These actions allow users to act on a record or in the context o that record. They appear in their own list view when a user clicks Related Actions in the left-side navigation of a record. Each related action has its own visibility setting to make it appear only to certain users and/or for certain records.

Active quick tasks for a process-backed record type are automatically added to the Related Actions list below any process model related actions.

By defining related actions for a record type, you can have data regarding a record pre-populate the action's process parameters and allow users to wc within the records interface.

NOTE: Only process models that have a Tempo-enabled start form or do not have a start form are supported as related actions. This restriction does not apply to related actions based on quick tasks.

To add a related action to a record type, access the Record Types tab from the Data Management page of the System view, select your record type, and click **Create a New Related Action** from the Related Actions box.

Fields

- *Target Process Model*: Process model to be added as an action.
- *Context Expression*: Expression used to pass details about a record to the related action's process model.
- *Visibility Expression*: The expression that determines the users and records for which the related action appears.

Notes

The context expression must return a dictionary where the keys have the same names as process parameters in the related action's process model. Required parameters must have a key in this dictionary or the process will not start. If any key in the returned dictionary cannot be mapped to a parameter for the specified process model, the related action will not execute. When the related action is launched, the key/value pairs in the context expression above map into the process parameters with the corresponding names which are then used to populate the start form (if one is configured)

You will have access to `rp!id` as well as the record fields under the `rf!` domain to use in the context expression.

For entity-backed records, `rp!id` is the primary key for the entity.

For process model records, `rp!id` is of type Process. If desired, you can cast this to an integer using the `tointeger()` function to get the process id.

For service-backed records, `rp!id` is whatever value was returned as the record's identifier from the source expression. The record's identifier must of o type Text in order for all aspects of the record to work. If the identifier is of another type, cast it to a string when necessary for service-backed records. For example, record tags for service-backed records will not work unless your record's identifier is of type Text or you use the `tostring()` function to ca the identifier.

If your process has a start form, only data in fields on the start form will be passed into the process when it starts. Data from the context expression is used to populate the form, but once the form values have been pre-populated, any other data from the context expression is discarded.

The visibility expression must return a boolean. Expressions that evaluate to "true" will result in the related action appearing for the logged in user. Expressions that evaluate to something other than "true" will result in the related action not appearing for the logged in user.

You will have access to `rp!id` as well as the record fields under the `rf!` domain to use in the visibility expression so that you can specify different visibility settings for records of the same type based on their attributes.

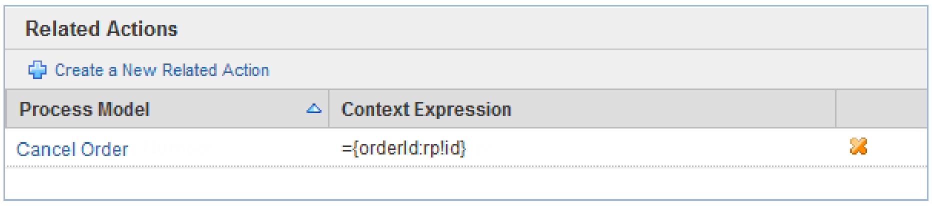
The visibility expression only determines the visibility of the related action in the Related Actions list; it does not determine the security of the underlying process model. The process model should be secured separately. Users who do not have permission to initiate a process model will not see the related action, regardless of its visibility expression.

Once you click **Create Related Action**, the related action is added to the list and saved to the record type. You do not need to click **Update Record Type** to save the new related action.

To add another related action, click Create a New Related Action and add new values for the fields above.

Examples

Related action for Canceling an Order on a Entity-Backed Record


Related Actions

Create a New Related Action

Process Model	Context Expression	
Cancel Order	={orderId:rp!id}	✗

Where `orderId` is the name of a process parameter in the process model named `Cancel Order`.

If your process model instead has a parameter of type Order, you might enter the following for Context Expression:

```
={order: rule!orderFromOrderId(rp!id)}
```

Editing Record Types

To edit a record type, complete the following:

1. From the Designer interface, select the **System** view.
2. From the list of Administration pages, expand **System Administration Home > Data Management**.
3. Click **Data Management**.
4. Select the **Record Types** tab.
   - A list of all existing record types displays in alphabetical order.
5. Select the record type you want to edit.
   - The Edit Record Type window displays.
6. Modify the values as needed.
   - If you modify the URL Stub, previously saved URLs no longer work.
7. Click **Update Record Type**.

To modify a default filter or facet for a process-backed or entity-backed record type, select it from the associated list and modify as needed, then click Update Filter/Facet.

To delete a default filter or facet for a process-backed or entity-backed record type, click the orange X next to the item in the associated list, and click Y to confirm.

To modify a related action, select it from the associated list and modify as needed, then click Update Related Action.

To delete a related action, click the orange X next to the item in the associated list, and click Yes to confirm.

Users can edit any rules in the record detail view or record dashboards without accessing the record type if they have Editor or Administrator rights to the rules.

The same applies for the record type's security groups. If they have Editor or Administrator rights to the groups, they can edit them without accessing the record type.

Version Control

Only the latest version of the record type is saved in the system.

Any changes you make to the rules created as part of the record detail view or record dashboards, however, are saved as a new version just as any other rule.

See also: Version Control for Rules

Deleting Record Types

Deleting a record type removes the object from the system. It will no longer appear in Tempo.

Any rules or security groups configured for the record type remain. Any process models or data store entities it references also remain.

To delete a record type, complete the following:

1. From the Designer interface, select the **System** view.
2. From the list of Administration pages, expand **System Administration Home > Data Management**.
3. Click **Data Management**.
4. Select the **Record Types** tab.
    - A list of all existing record types displays in alphabetical order.
5. Click the orange **X** next to the record type you want to delete.
    - A confirmation window displays.
6. Click **Yes**.

Design Best Practices

Record Definition

Save Record Configuration Expressions as Rules

Appian recommends saving any expressions created for a record type as a rule for version control and testing purposes. This also allows you to re-use the rules for a different record type.

Pass Record Data for Rules as Rule Inputs

The `rf!` and `rp!` domains are not available from within the rules management interface. This means that if you try to test a rule that references `rf!`, the rule will not evaluate. It is better to pass all of the record data the rule needs into the rule as inputs.

For instance, a rule for a SAIL Summary Dashboard should look like the following, where the `getMyDashboard` rule defines one rule input each for the `customerName`, `industry`, and `contactInfo` field:

```
getMyDashboard(rf!customerName, rf!industry, rf!contactInfo)
```

Create Constants for your Record Types

If you save your record type as a constant, you can use it as a parameter value for functions related to records, such as `queryrecord()` and `urlforrecord()`.

See also: queryrecord(), urlforrecord(), and Constants

Record Performance

Avoid Using Slow Expressions

When a user accesses the record list view, the record list view item template expression must evaluate for every record in the record list view before the list view can render.

Accessing the data you wish to display using in the list view using `rf!` helps the expression execute faster than it would if the data were accessed by executing a query rule or otherwise looking up data. Appian recommends adding fields to your record for the data you wish to display in the list view rather than performing a look-up.

Record List View Design

Provide High-Resolution Images for Record Icons

Appian will automatically optimize the image for display on web and mobile clients that support a variety of screen resolutions and pixel densities. Manual reducing the source document size may result in certain users seeing low-quality icons with improper layout.

Use Distinct Images for Records

When selecting the document source for the records images, make sure it results in images that help users identify each record easily. For example, usi a company's logo as the image for each record in a Companies record type helps the user find a company by glancing at the record list view. Using the same generic icon for all listed companies would not. Similarly, you should not use a user object for the record list view image unless that record represents the person or is very strongly tied to that person.

Create a Default Filter for Deleted Records

If the source has existing instances that are flagged as deleted (such as rows of the data store entity or processes of the process model), consider creating a default filter to hide them from the record list view.

Create Facets to Show Additional Records

Only the first 100 records display on a record list view. If the number of records for a record type exceeds 100, create simple alphabetical facets so that each option displays up to 100 records.

Record Usability

When users visit the Records tab, they'll be primarily focused on finding a particular record and specific information that pertains to it. Follow the best practices below to make it easiest for users to scan through your records.

Make Sure Record Titles are Unique

Adding repetitive prefixes, such as categories, to record titles only adds noise to the record title and doesn't help users locate the specific record they're looking for.

Keep Record Titles Concise

Long titles, especially those wrapping onto multiple lines for mobile applications, make it harder for users to quickly browse through the titles.

Avoid Using Complex Expressions For Record Titles

If record titles are defined by complex expressions that evaluate differently for one user over another or include metadata (such as a priority level or tas count), it makes it difficult for users to discuss or search for the same record without confusion over the title. Specific functions to avoid include `loggedinuser()` and `if()`.

Define Concise Record Descriptions to Make Record Lists Easy to Scan

Long, paragraph-style, descriptions will reduce the number of records that are visible on the screen at the same time. They are also harder to read whe quickly scanning down the record list than either a shorter sentence-style description or a series of label/value pairs separated by line breaks.

Specify a Timestamp Only When it is Intuitive and Useful

The timestamp for records on a record list view is optional. Since the timestamp is not labeled, one should only be shown if users are likely to understan its meaning without further explanation (for example, when an event represented by the record occurred). If there is a requirement to show timestamp which require explanation, include them on the record dashboards where they can be shown with explicit labels.

Use Appropriate Column Layouts for Record Dashboard Content

To create visually-balanced record dashboards, it is recommended to use the same layout (one-column or two-column) for all sections. Two-column layouts provide greater density for short text values, smaller charts, and grids. One-column layouts provide additional space to show longer paragraphs of text, as well as charts and grids that contain more data points. A one-column layout is recommended for grids that include more than 5 columns and/or lengthy text content. Charts that show more than 7 data points are generally best-shown in one-column layouts.

When including multiple sections on a dashboard, make sure that the height of content in each column of two-column sections is similar in order to minimize white space.

When mixing different layouts on the same dashboard, it is preferable to place a one-column section above a two-column section; this reduces the likelihood of a shorter column creating empty space above the start of the next section. An example of an effective mixed-layout dashboard is a grid in one-column section above a two-column section containing a variety of brief text and number data values.

Keep Label Text Short

Use concise labels to describe record dashboard fields. Most of the space within the dashboard should be used to show data values; avoid having verbo labels distract from content. If detailed description of a value is needed for proper comprehension, provide instruction text instead of a lengthy label.

Use Sections to Organize Content

Group related fields into sections described by concise labels. This helps users to scan dashboards to find relevant information and also reduces the need for redundant text in field labels.

Troubleshooting

Why are My Record Images Not Displaying Correctly?

If an image on a record list view displays with the wrong colors or doesn't display at all, make sure the file extension matches the file type the image was originally saved as. For example, if an image is saved as a PNG file and the extension is changed by modifying the file name (such as renaming `image.pn` to `image.jpg`) before it is uploaded to Appian, the image will not display correctly on a record list view.

See also: Adding and Sharing a File

See Also

Records Tutorial: Walks you through an example of creating your first record.

SAIL Tutorial: Walks you through an example of creating your first SAIL, which is the basis of a record dashboard.

SAIL Components: Lists the supported SAIL components and the data structure required for adding them to a record dashboard.