Tutorials

Appian Academy

Interested in learning the fundamentals of Appian through free, online, video courses? Click the link above to get started.

The tutorials available here are best completed in the following order:

Application Building

Create a basic application with an action. This tutorial is a great place to start if you are new building applications and securing design objects.

Interface

Create a basic interface. This tutorial is a great place to start if you are new to designing interfaces in Appian.

Process Modeling

Create a process that allows users to submit expense reports.

Task Report

Create a task report that displays the tasks from the *My Tasks* process report.

Entity-Backed and Process-Backed Records

Advanced tutorial that walks you through the process of creating an entity-backed record and process-backed record.

Expression-Backed Records

Advanced tutorial that walks you through the process of creating a expression-backed record.

Grid

Create a read-only grid using the Report Builder.

Web API

Advanced tutorial that walks you through creating a Web API that retrieves a JSON-encoded list of records.

Web API - Level II

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Advanced tutorial that walks you through creating a Web API that creates a new entry in a Data Store.

Design Objects

Overview

Applications contain a set of objects that function together to meet one or several business use cases. Applications allow these objects to be transported from one environment to another.

Each design object provides a specific piece of functionality, and each application comprises many objects grouped by common purpose. Technically, Applications do not contain these objects, but simply have a list of objects that are associated with them. The objects view shows a list of objects ignoring the Application association. The following sections describe each Appian object.

Data Objects

Records, Data Stores and Custom Data Types are all data-centric object. Their icons will appear orange in the Application and Object view.

Data Type

As opposed to a primitive or system data type a custom data type (CDT) is a designer-defined data structure. DTs allow designers to create a logical grouping of related data that can then be used by other objects to share data. Data can be shared internally, for instance between an Interface and Process Model, or externally, between an Expression Rule and Data Store.

Because a data type will always be used in context of another object, it does not have individual security settings.

Data Store

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

A data store is a connection to an external relational database that is used to store application data. Each data store contains one or more data entities. When saving data from Appian to an external database, the data store defines the connection to the database, while data types define the structure of the data entity being stored.

Record Type

A record type brings together all the data on a single topic and displays it in a series of record views. Records provide a centralized view of a given business function, along with all of its connections to related records.

Attaching process models to record views as related actions allows users to immediately take action on the information shown in the record view.

Process Objects

Process Models and Process Reports can be considered process centric. Process Models define how a process will function while Process Reports allow users access to data from the process.

Process Model

A process model is the primary tool in Appian for describing a workflow. Designers graphically lay out the workflow, which may assign user tasks, manipulate data, post system events, or update other design objects. Process models are frequently used with record types to provide users with tools to act on the information shown by the record.

Process Report

A process report displays data from active and recently completed processes and tasks. Designers can choose to create process reports from scratch or pick from one of several dozen of out-of-the-box report templates.

User Objects

Objects that are primarily involved with interactive user displays fall into this category. Interfaces, Report, and Sites are all objects are created in order for users to interact with the Application. Although, Record Types provide user interaction functionality, the object also is a query-able source of data.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Interface

An interface is an object that returns one or more components to display on a record view, Tempo report, or process form. This is the primary object that designers use to show user interfaces to application users.

Report

A report displays data from tasks, records, and other data sources in a single interface in Tempo for end users to view. Through the use of graphs, grids, pictures, and the dynamic behavior that [SAIL][SAIL_Design.md] offers, a report offers a high-level overview of aggregated data.

Site

A site is a customizable user interface where Designer can create focused work environments for their user. When working in a site, users can view a task report, submit tasks, and kick of new actions, all without seeing the five tab navigation bar shown in Tempo.

Rule Objects

Rule based objects are used in Expression to reference specific values and perform complex operations or queries. Expression Rules, Decisions, Constants, and Query Rules are all considered rule-based objects.

Expression Rule

An expression rule is a stored expression that works like an Appian function, except that users can create their own rule inputs (to use as parameters) and definition.

Like all expressions, an expression rule is a statement that evaluates to return a value, much like a spreadsheet function.

Decision

A Decision is a grouping of business rules that answers a specific question based on inputs. Unlike expression rules, which primarily calculate or manipulate data, Decisions are best used to encapsulate complex, business-specific logic.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Decisions can be called from any expression, so they can be reused across multiple objects throughout the system.

Constant

A constant holds a single user-defined value or list of values. A constant allows you to define a value once and then use it in many places in an application. If the value needs to be updated in the future, it only needs to be updated in one location. Constants are also used to reference other design objects in expressions. Common uses for constants include:

- the number of days allowed for approval of a task
- the standard label for all Submit buttons in the application, or
- a reference to the Customer record to use in the 'queryrecord()' function

Query Rule

A query rule is a defined query against a single data store entity. It returns an array of the values that match the parameters defined in the query rule. Designers may also use the a!queryentity() and queryrecord() system functions to perform similar queries.

Integration Objects

Whenever an application needs to interact with a third-party system or vice versa, designers will use as many as three different type of objects: an Integration, a Connected System, and a Web API

Integration

An integration can be used to call external systems and web services from Appian. Integrations can be called in expressions, interfaces, and process models to query or modify data in external systems. They can inherit connection details and logo images from a connected system.

Connected System

A connected system represents an external system or service that is integrated with Appian, for example, Microsoft Sharepoint, MuleSoft, or a custom web service. Connected systems have logos which can be customized to help visually identify the system or service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

A connected system allows you to define the details of a connection once and then use those details in one or more integrations. With a connected system, if a value needs to be updated in the future, for example if a password changes, it only needs to be updated in one location. Connected systems also make it easy to manage connections that change as you deploy integrations from development to testing and then production.

Web API

A Web API provides a way to expose data that is stored inside of Appian or is accessible by Appian to another system. Each Web API is an association between a URL and an expression. When a client makes an HTTP request to the given URL, the associated expression is executed and the result is returned to the client. This means that any data that is available inside an expression can be exposed to an external system via a Web API.

Group Objects

Appian managed object permissions through group membership and system roles. Groups and Group Types are the objects that support security and permissions throughout the Application

Group

A group allows designers to organize users, usually for the purpose of determining what permissions they have to design or use application objects and data. In addition, tasks and News entries can be targeted to one or more groups, as well as to individual users. Every group has a group type (defined below), as well as a list of users and member groups that belong to it.

Group Type

A group type is used to organize groups, and can only be created by users of type System Administrator. For example, the Region group type allows a designer to organize their sales teams by creating a different Region group for each sales team, for instance, Commercial West, Commercial East, Midwest.

Group types also define attributes that are shared across groups. For example, the group type Region might have a "regional VP" attribute. Then, each group of that group type would have a different value for the attribute, based on who that region's vice president is.

Content-Management Objects

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Appian has a robust content management framework that allows designers to store and organize Application content. For document management, there are three specific objects involved: Knowledge Centers, Document Folders, and Documents. Additionally, process model and rule folders exist to assist in the organization of these objects.

Document

A document is a file stored in Appian. Appian provides a management system for documents.

While process reports are stored as documents, they have a unique icon and are sorted and filtered as reports.

Folder

Folders allow you to organize your application content and centralize security settings. Design objects and documents can only belong to one folder at a time.

You can create folders within folders for multiple levels of organization.

Items in a folder (including sub-folders) are listed in both the application view and the folder view.

Security settings for the folder apply to all items within it (with the exception of Process Model Folders).

Folders created within other folders by default inherit security from their parent folder. Edit the security of the child folder to change this option: **Inherit security from parent**.

There are four types of folders to choose from (**Rule**, **Process Model**, **Document**, and **Knowledge Center**).

Rule Folder

Rule folders can store the following design object types:

- Constant
- Expression Rule
- Interface
- Query Rule
- Rule Folder

Process Model Folder

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Process model folders can only store process models or other process model folders.

Security set for a process model folder is not applied to its contents.

Document Folder

Document folders can contain documents or other document folders.

Document folders can only be created within a knowledge center or within another document folder.

Knowledge Center

A knowledge center stores documents and document folders.

Notification Objects

A feed object is created to support notifications on News in Sites or Tempo.

Feed

A feed is a channel for delivering content to the News Feed in Tempo or Sites. Every post or event in the News Feed that isn't directly created by a user has a feed associated with it. Generally, designers use a separate feed for each topic for which their application creates events or comments.

Appian Data Types

Overview

The data (process variables, node inputs, node outputs, rule inputs, data store entities, or constants) used by Appian must conform to certain data types.

Appian data types can be one of the system types or a custom data type built from an XML Schema Definition (XSD), Java object, or imported from a WSDL by the Call Web Service smart service.

See Also: Conversion Functions

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Any Type

The Any Type is a generic data type only available for use as an expression input for rules and certain expression functions, such as the **if()** function. It accepts data of any data type.

Data is stored in variables of this type by mapping an existing variable, rule, constant, or expression to its value.

Primitive System Data Types

A system data type is a required format for data stored in Appian and includes primitive types and complex types. Each system data type can be used to store either a single value or a list of values.

The following primitive system data types are available.

Boolean

Values include **True / False**.

- **1** is accepted as a literal value meaning Yes.
- **0** is accepted as a literal value meaning No.

The **default** value is null, which appears as `[Empty Value]` in a process variable. The output of an empty boolean value in an expression depends on the function used.

You can populate the value using results from the `true()` or `false()` functions.

The **toboolean()** function can be used to convert `true`/`false` and `0`/`1` values into a Boolean data type value.

See also: Casting

Date

Data in a date format can be created using the date(year, month, date) function. Variables that have a **date** data type do not accept text string input.

The **default** value for a date is [Empty Value]. The **minimum** value is 1/1/1000, and **maximum** value is 12/31/9999. Dates values are *not* adjusted to a different time zone when saved or displayed.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Numerical and text values can be converted to a date value using the **todate()** function.

Date and Time

Variables that hold a Date and Time data type refer to a point in time that is the same for all users. Date and time variables do not accept text strings as input.

A Date and Time value is saved in Greenwich Mean Time (GMT) then converted to the end-user's time zone (accounting for daylight saving time) when displayed.

GMT is the default time zone used when evaluating expressions that include Date and Time values. You can specify a different time zone in your process model properties. The time zone used for display can be the user's preferred time zone, a globally specified time zone, or the time zone context.

If a function expects a Date and Time value, and you pass it a date, the date is automatically converted to a Date and Time value, using 12:00 am for the time component.

The Date and Time value is only converted to the end user's time zone when it is displayed in the following manner. (Use separate date values and time values if you do not want this conversion to take place.)

- As a user input
- When used in a calendar function
- When cast to a string

See also Time Zone Context

Date and Time format data can be created using the **datetime(year, month, date, hour, minute, second)**function.

Numerical and text values can be converted to a Date and Time value using the **todatetime()** function.

See also: Appian Functions, Internationalization Settings

Encrypted Text

This type is used to store an Encrypted Text value. An Encrypted Text value can be created in one of the following ways:

- Entered by a user in a EncryptedTextField component
- Generated in a plug-in using the `EncryptionService` public Java API

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

An Encrypted Text value is only decrypted when displayed as the value in an EncryptedTextField or within a plug-in using the `EncryptionService` public API.

A value of type Encrypted Text cannot be cast to any other type and no other type can be cast to it.

An encrypted value is larger than the corresponding plaintext value. Specifically, the value stored in memory or on disk will be the lowest multiple of 16 bytes that is greater than the size of the corresponding plaintext value in bytes.

An Encrypted Text value remains encrypted when stored on disk. The encryption key is unique to each installation.

The data type does not provide any additional access controls. Encrypted text entered by one user may be decrypted and displayed to another user if that other user has permission to view the interface in which the value is displayed.

See also: Encrypted Text Component

Number (Decimal)

Holds numeric data stored as double precision floating-point decimal numbers. The default value is **0.0**.

Decimal numbers can be created from text strings using the **todecimal()** function.

For forms, use one of the following Text Functions to display a decimal number as a default value to avoid rounding.

- fixed()
- text()

Also for forms, use one of the following currency functions to display a decimal number as a default currency value.

- dollar()
- euro()
- pound()
- yen()
- currency()

See also: Text Functions

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

For PVs, if you enter a number that exceeds the maximum number of digits supported by double precision floating-points, the number is truncated down to the maximum number of digits when you save the process model. It does not provide you with a warning message if this occurs.

Number (Integer)

Integer numbers can range from `-2,147,483,647` to `2,147,483,647` (or from `-231+1` to `231-1` in scientific notation).

The default value is `0` and the null value is `-2\^31`.

Integer numbers can be created from text strings using the **tointeger()** function.

When an arithmetic operation (such as an expression) creates a Number (Integer) value that exceeds the type's limits, the value wraps.

- `2147483647 + 10 = -2,147,483,639`
- `2147483647 + 1 = -2147483648` - interpreted as null. When looking in the user interface at a process variable changed to this value, an [Empty Value] result is displayed.

When values that exceed the Number(Integer) range are passed through a user interface, the excessive value is changed.

- In most cases, this is changed to a null value.

When a value that exceeds the Number(Integer) range is converted to Number(Integer) from a string in an engine server (such as when you convert text to an integer using the **tointeger()** function), the excessive value is changed to the maximum value.

On task forms, the Number Form Component prevents entry of values outside of the valid range.

- If an expression is entered for a value and the Number Form Component is not mapped to a node input, any values that exceed 7 digits display in standard formatting with 7 digits of precision. For example, `2147483647` displays as `2147483000`.

For PVs, if you enter a value that exceeds the Number(Integer) range, the integer will be replaced with a null value when you save the model. The Modeler does not provide you with a warning message if this occurs.

Text

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

This type is used to store any UTF-8 text string. Numerical values can be entered into the text data type; however, data manipulation cannot be performed on the text data type (except for report aggregations). The default value is [Empty Value].

To display text, enclose it within double quotation marks ("").

Time

Time data can be created using the **time(hour,minute,second)** function. Variables that have a time data type do not accept text string input.

Time values are *not* adjusted to a different time zone when saved or displayed.

Complex System Data Types

The following complex system types are made available in the system to support smart services.

These types cannot be edited or deleted. Their XML structure is not guaranteed to remain the same from release to release.

ApplicationTestResult

The ApplicationTestResult data type is designed to hold test result information for all expression rules in an application. This type also includes execution statistics for an individual application.

See also: ApplicationTestResult and Automated Testing for Expression Rules

DataSubset

The DataSubset data type is designed to hold the data returned by a query rule configured with a paging parameter.

It contains the following fields:

- **startIndex** - This field holds a single Number(Integer) record.
- **batchSize** - This field holds a single Number(Integer) record.
- **sort** - This field holds multiple SortInfo records.
- **totalCount** - This field holds a single Number(Integer) record.
- **data** - This field holds multiple Any Type records.
- **identifiers** - This field holds multiple Any Type records.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

EntityData

The EntityData data type lets you define a target data store entity and the values to store in the target entity as an input value for the **Write to Multiple Data Store Entities Smart Service**.

It contains the following fields:

- **entity** - This field holds a single Data Store Entity value in which the data to be updated is stored.
- **data** - This field holds multiple Any Type values to store in the entity.

For example, a value of type EntityData where the entity and data values are stored as process variables could resemble the following when using the dictionary syntax:

```
{entity:pv!ENTITY_OPPORTUNITIES, data:{pv!RadiationOpp, pv!NewBusinessOpp}}
```

See also: Data Store Entity Data Type and Write to Multiple Data Store Entities Smart Service

EntityDataIdentifiers

The EntityDataIdentifiers data type lets you define a target data store entity and the values to delete from the target entity as an input value for the **Delete from Data Store Entities Smart Service**.

It contains the following fields:

- **entity** - This field holds a single Data Store Entity value in which the data to be deleted is stored.
- **identifiers** - This field holds multiple Any Type values for the primary key values of the data to be deleted.

For example, a value of type EntityDataIdentifiers where the entity and data values are stored as process variables could resemble the following when using the dictionary syntax:

```
{entity:pv!ENTITY_OPPORTUNITIES, identifiers:{pv!RadiationOpp.id, pv!NewBusinessOpp.id}}
```

NOTE: Make sure to use the primary key value IDs for the data rather than CDT values (for example, `pv!opportunities.id` rather than `pv!opportunities`). Using a CDT value will result in a casting error.

See also: Data Store Entity Data Type and Delete from Data Store Entities Smart Service

LabelValue

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

The LabelValue data type is designed to hold event labels and a nested hierarchy of sub-event labels for use by the **Post Event to Feed Smart Service** and the **Post System Event to Feed Smart Service**.

It contains the following fields:

- **label** - This field holds a single Text record.
- **value** - This field holds multiple Any Type records.

LabelValueTable

The LabelValueTable data type references the LabelValue type.

It contains the following field:

- **LabelValue** - This field holds multiple LabelValue records.

ListViewItem

Data type used to define the record list view for record types.

It contains the following fields:

- **image** - This field of type Document or User defines the image to appear in the list view next to each item.Value must be entered as an expression. If left null or empty, the first two letters of the record title display. For image file types, a thumbnail of the document displays. For user values, the user's avatar displays.
- **title** - This field of type Text defines the name or short text description of the item.
- **details** - This field of type Text defines a longer text description of the item.
- **timestamp** - This field of type Date and Time indicate the creation modification timestamp of the item. Valid values include variables for timestamp fields of the record such as creation timestamp, a last modified timestamp, or other timestamp.

See also: a!listViewItem()

Facet

Data type produced when defining user filters using expressions.

It contains the following fields:

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

- **name** - A Text value defining the name of the user filter that displays to end users.
- **options** - An array of FacetOption values defining the list of options that a user can select from. See below: FacetOption

See also: a!facet()

FacetOption

- **id** - An Integer value defining the unique identifier for the filter option.
- **name** - A Text value defining the name of the option.
- **filter** - The QueryFilter value that will be sent by the framework when this filter option is selected. See below: QueryFilter
- **dataCount** - An optional integer value defining how many items in the data set will be selected if this filter option is chosen.

See also: a!facetOption()

ObjectTestResult

The ObjectTestResult data type is designed to hold data for each of the expression rules with test cases.

See also: ObjectTestResult and Automated Testing for Expression Rules

PagingInfo

The PagingInfo data type is designed to hold the paging configuration passed as a parameter to query rules and the Paging Grid component.

It's used primarily as an argument for the `todatasubset()` and `a!gridField()` functions.

To create a value of type PagingInfo, use the `a!pagingInfo()` function.

See also: todatasubset() and a!pagingInfo()

ProcessInfo

The ProcessInfo data type is designed to hold information about a running process. It contains three fields.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

- **pp** - A dictionary containing the properties of the process: id, name, priority, initiator, designer, start time, deadline, and timezone
- **pm** - A dictionary containing the properties of the process' model at the time the process was started: id, name, description, version, creator, and timezone
- **pv** - A dictionary containing the process variables of the process

See also: Process Model Properties, Process Variables

Query

The Query data type defines the grouping, aggregation, filtering, paging, and sorting configuration to be applied when querying record data. To do so, you use it as an input to the queryrecord function to return a DataSubset value containing query results and must be created ad-hoc via a type constructor. The DataSubset can then be applied to an interface component, such as a grid, pie chart, or stacked bar chart.

NOTE: Process variables cannot be created as a Query data type.

It contains the following fields:

- **selection|aggregation** (Selection or Aggregation) - This optional field determines the selections or grouping and aggregations for the query. Only one Selection or Aggregation value can be used. If neither are provided, all fields of the record type are returned.
    - See also: a!querySelection(), a!queryAggregation(), Selection and Aggregation
- **logicalExpression|filter|search** (LogicalExpression, QueryFilter, or Search) - This optional field determines the filtration to apply to the query. Similar to the selection|aggregation field, only one value can be used. To include more than one filter, use the LogicalExpression data type with the AND operator. If none of them are provided, no filters will be applied.
    - See also: a!queryLogicalExpression(), a!queryFilter(), LogicalExpression, QueryFilter, and Search
- **pagingInfo** - This required field holds a PagingInfo data type value and determines the paging configuration to use.
    - See also: a!pagingInfo(), PagingInfo

Validation of the Query data type occurs when it is evaluated with the `queryrecord()` function.

See also: a!query(), Record Design, and queryrecord()

Selection

Data type accepted in the **selection|aggregation** field of the Query data type. It can contain one or more Column data types and should be used instead of an Aggregation data type when you just want to select the columns, rather than group them together or apply an aggregation function.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

See also: a!querySelection()

Column

The Column data type is only used in conjunction with the Selection data type.

It contains the following fields:

- **field** - The field of the data type you want to retrieve. The fields available depend on the source of the data and the data type of that source. Fields that are children of a multiple cannot be selected. If the alias is not provided and the field name collides with another existing alias, the field name will be suffixed with an incremented digit appended to the end when returned in the result.
    - See also: Record Design
- **alias** - (Optional) The short name by which the result of the Column value can be referenced in other areas of the query value. Values are case-sensitive. If no alias is given, the alias for the column will be inferred as the *field* value.
- **visible** (Boolean) - (Optional) Determines whether the column should be visible to end users. If false, the data for the column will not be retrieved, but it can be used for sorting. Default value `true`.

See also: a!queryColumn()

Aggregation

Data type accepted in the **selection|aggregation** field of the Query data type. It can contain one or more AggregationColumn data types and should be used instead of a Selection data type when you want to perform a function on the selected columns. The following aggregation functions are supported: `COUNT`, `SUM`, `AVG`, `MIN`, and `MAX`.

See also: a!queryAggregation()

AggregationColumn

The AggregationColumn data type is only used in conjunction with the Aggregation data type.

It contains the following fields:

- **field** - The dot-notation to the field of the data, such as a record type, you want to group together and/or aggregate. The fields cannot be complex or multiple values.
- **alias** - The short name by which the result of the AggregationColumn value can be referenced in other places of the Query value. Values are case-sensitive.
- **visible** (Boolean) - (Optional) Determines whether the grouping or aggregation column should be visible to end users. If false, the data for the column will not be retrieved, but it can be used for sorting. Default value is `true`.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

- **isGrouping** (Boolean) - (Optional) Determines whether the field should be grouped. Default value is `false`.
- **aggregationFunction** - The function to use when aggregating the field. Valid values include COUNT, SUM, AVG, MIN, and MAX. This value is required when *isGrouping* is set to `false`.
- **groupingFunction** (Text): A function that can be applied on the selected field. Valid values are YEAR and MONTH. This parameter can only be used with `Date` and `Date and Time` data types. Requires *isGrouping* to be `true`.

See also: a!queryAggregationColumn()

LogicalExpression

Data type that determines the filtration to apply.

It contains the following fields:

- **operator** - Determines the operation to apply to the set filters in the *logicalExpression|filter|search* value. Currently the only valid values are `AND` and `OR`.
- **logicalExpression|filter|search** (LogicalExpression, QueryFilter, or Search) - Nested LogicalExpression or QueryFilter values that will be operated on based on the *operator* value.

See also: a!queryLogicalExpression()

QueryFilter

This data type is required to configure the filter options for a expression-backed record or filter a `queryrecord()`function call before any grouping or aggregation is computed.

It contains the following fields:

- **field** - The dot notation to the field that you want to apply the filter to.
- **operator** - The operator to apply to the filter. Valid values include `=`, `<>`, `>`, `>=`, `<`, `<=`, `between`, `in`, `not in`, `is null`, `not null`, `starts with`, `not starts with`, `ends with`, `not ends with`, `includes`, `not includes`.
- **value** - The value to compare to the given field using the given operator. Optional if the *operator* value is `is null` or `not null`. If the *operator* value is `between`, the value must be a list of only two elements with the lower bound as the first element and the upper bound as the second.

See also: a!queryFilter(), queryrecord()

Search

Data type that indicates a user's search term. Used in the **logicalExpression|filter|search** field of a Query or LogicalExpression data type.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

It contains a single field:

- **searchQuery** - The text value to look for.

SortInfo

The SortInfo data type is referenced by the PagingInfo and DataSubset types and determines how data is sorted in a subset.

To create a value of type SortInfo, use the `a!sortInfo()` function.

See also: a!sortInfo(), PagingInfo, and DataSubset

TestCaseResult

The TestCaseResult data type is designed to hold data for each of the test cases in an object.

See also: TestCaseResult and Automated Testing for Expression Rules

TestRunResult

The TestRunResult data type is designed to hold test statistics for a test run across all applications being tested.

See also: TestRunResult and Automated Testing for Expression Rules

Writer

The Writer data type is a special data type returned by expression functions that intend to modify data. The modification of data must not happen during expression evaluation, so these functions return a Writer, which is then handled in a special way during the phase of an interface evaluation where saving into variables takes place. The Writer data type has no impact during expression evaluation - no data is written by the function that returns a writer until a variable created with the bind function is saved into in an interface.

It contains the following fields:

- *name* - The name of the function that returned the writer
- *parameters* - The parameters that will be used when the writer executes the data update

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

See also: bind() and Writer Functions

Appian Object Data Types

An Appian Object data type is a required format for objects specific to the Appian system. Similar to primitive and complex system types, each can be used to store either a single value or a list of values.

Appian Object data types are only recognizable within the Appian system.

Application

Holds an integer ID number that represents an Appian application. It can be used as a rule input to expression rules or interfaces; it can also be used as a constant and referenced from interfaces, web APIs, and process models; and finally used as a process variable from the process modeler, or as inputs to the Start Rule Tests (Applications) - Smart Service. Custom data types cannot use this data type.

Connected System

Holds an integer ID number that represents a connected system. A connected system represents an external system that is integrated with Appian.

See also: Connected System Objects

Data Store Entity

Holds an integer ID number that represents a data store entity. Data store entities are named, typed storage units within a data store. They can map to one or more tables in an external database.

It can only be applied to process variables and cannot be used to save form data from a mobile form. Data store entity IDs are not reused if the entity is deleted.

See also: Data Stores

Document

Holds an integer ID number that represents a document in Document Management. It can be used to save form data selected from a dropdown, radio button, or checkbox field input on a mobile form.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

If the ID number of a document is known, you can convert it to this data type using the **todocument()** function. Document IDs are not reused if the document is deleted.

See also: Document Management

Document Management Community

Holds an integer number that represents a Document Management Community ID.

If the ID of a Community is known, you can convert it to this data type using the **tocommunity()** function. Community IDs are not reused if the community is deleted.

It cannot be used to save form data from a mobile form.

Document or Folder

Holds an integer ID number that represents a document or a folder that exists within Document Management. Document or folder IDs are not reused if the object is deleted.

It cannot be used to save form data from a mobile form.

Email Address

Holds data formatted as an email address. Variables that hold it do not accept text strings as direct input.

Email address data can be created from text strings using the **toemailaddress()** function.

Use the email recipient data type if you are sending email from a process.

It cannot be used to save form data from a mobile form.

Email Recipient

Email address data must be converted to email recipient data for use by the Send Email smart service.

This is done with the **toemailrecipient()** function, which accepts email address data, user data, or group data.

It cannot be used to save form data from a mobile form.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Folder

Holds an integer ID number that represents a folder that exists within Document Management. It can be used to save form data selected from a dropdown, radio button, or checkbox field input on a mobile form.

If the ID number of a folder is known, you can convert it to this data type using the **tofolder()** function. Folder IDs are not reused if the folder is deleted.

Group

Holds an integer ID number that represents a group within the system.

It can be used to save form data selected from a dropdown, radio button, or checkbox field input on a mobile form.

If the ID of a group is known, you can convert it to this data type using the **togroup()** function. A group ID may be reused if the group is deleted.

See also: Creating Groups

Knowledge Center

Holds an integer ID number that represents a Knowledge Center in Document Management.

If the ID number of a Knowledge Center is known, you can convert it to this data type using the **toknowledgecenter()** function. Knowledge Center IDs are not reused if it is deleted.

It cannot be used to save form data from a mobile form.

Process

Holds the integer ID number of an instance of a process model.

Process Model

Holds the integer ID number of a process model. Each time a process model is launched, it runs as separate process.

A process model ID may be reused if the process model is deleted.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Record Identifier

Holds the definition of a record.

You can only create constants of this data type. Process variables cannot be created of this type, and custom data types cannot be saved as this type.

To create a value of type Record Identifier, use the `a!toRecordIdentifier` function. To create a value of type Record Identifier for a User record, use the `a!userRecordIdentifier` function.

See also: a!toRecordIdentifier() and a!userRecordIdentifier()

Values of type Record Identifier are used as inputs for the Post Event to Feed Smart Service and Post System Event to Feed Smart Service to specify the record tags for an event.

See also: Post Event to Feed Smart Service and Post System Event to Feed Smart Service

RecordType

Holds the definition of a record type.

You can only create constants of this data type. Process variables cannot be created of this type, and custom data types cannot be saved as this type.

Constants of type RecordType are commonly used as parameter values for functions related to records, such as `queryrecord()` and `urlforrecord()`.

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

Report

Holds an integer ID number that represents a report within the system.

Constants of type Report are commonly used as parameter values for report links.

SafeURI

Holds a URI value and enforces security rules during casting.

When casting from Text, the string will fail verification if it contains any of the following:

- Any scheme other than `http`, `https`, `ftp`, and `mailto`

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

- Invalid URI characters if not already escaped
- Empty text string

The string does not change when cast to or from a Text data type.

This data type can not be used in expressions for events or process reports.

See Also

Safe Link: Link type that accepts SafeURI values to create an external link.

Task

Holds the integer ID number of a process task.

It can be used to add a link to the Paging Grid component that opens a process task in Tempo.

See also: Paging Grid

Task Report

Holds an integer ID number that represents a task report within the system.

Constants of type Task Report are commonly used as parameter values for report links.

User

Holds an Appian user account ID number.

It can be used to save form data selected from a dropdown, radio button, or checkbox field input on a mobile form.

User or Group

Holds an Appian user account or group. It is sometimes referred to as a **People** data type.

It can be used to save form data selected from a dropdown, radio button, or checkbox field input on a mobile form.

Custom Data Types (CDTs)

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Designers can create or import their own custom data types (CDTs). These organize data into a structure that represents a logical grouping of related data, such as Employee and Contract.

For more information about creating and editing data types, see Custom Data Types (CDTs)

Mapping Data

Careful attention must be paid to data types when passing the values from one variable to another (also called mapping data).

The variables used at the node level are called node inputs and node outputs. (A node is often referred to as an activity. A node input or output can also be called an Activity Class.) These variables can be mapped into process variables, enabling process modelers to access the variables in other nodes within the process model and pass the data to other processes and sub-processes. The following properties must be taken into consideration when mapping node inputs/outputs to process variables.

- Appian does not support a direct mapping of data from one type into another, regardless of whether the data values are compatible. You can use an expression to cast a variable from one type into another and then save the result into a new variable.
- Appian allows you to map variables that only store single values into variables that can store multiple values using a custom output on the output tab, but not when using results listed on the output tab. Results must match according to data type and whether the data type holds multiple values (its cardinality).
- Mapping scalars (single values) to vectors (multiple values) is only applicable when mapping process variables. Therefore, vector values cannot be set as default values in a scalar variable. For example, a variable cannot be given the default value {1,6,9,8} if the variable does not support multiple values.

Data types also apply to rule inputs and constants, but data from a node input or a process variable cannot be mapped to a rule input or a constant.

See also: Casting

Interface Components

Interfaces in Appian are made up of components. This page lists all the components and supporting configurations delivered with Appian.

- If you're looking for information about the **PATTERNS** tab of the component palette, see Interface Patterns.
- To understand the Interface design, concepts, and functionality available, see Interface Object.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

- If you're new to Appian, check out Academy Online.

Layouts

Billboard Layout - Displays a background color, image, or video with optional overlay content.

Box Layout - Displays any arrangement of layouts and components within a box on an interface.

Card Layout - Displays any arrangement of layouts and components within a card on an interface. Can be styled or linked.

Columns Layout - Displays any number of columns alongside each other. On narrow screens and mobile devices, columns are stacked.

Dashboard Layout - Displays any arrangement of layouts and components. Use this as the top-level layout for record views and Tempo reports.

Form Layout - Displays any arrangement of layouts and components beneath a title and above buttons. Use this as the top-level layout for start and task forms.

Section Layout - Displays any arrangement of layouts and components beneath a section title on an interface.

Side By Side Layout - Displays components alongside each other.

Layout Elements

Column Layout - Displays a column that can be used within the columns layout.

Side By Side Item - Displays one item within a side by side layout.

Inputs

Barcode - Displays a field that allows entry of a barcode value using a barcode scanner or manual input.

Date - Displays and allows entry of a single date (year, month, day). When the field is editable, users can input dates by typing or by picking from a calendar.

Date and Time - Displays and allows entry of a single date and time (year, month, day, hour, minute, second). When the field is editable, users can input dates by typing or by picking from a calendar and then select the time from a dropdown.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Decimal (Floating Point) - Displays and allows entry of a single decimal number, stored with a floating point representation.

Encrypted Text - Allows entry of a single line of text that is encrypted when saved into a variable.

File Upload - Allows users to upload a file.

Integer - Displays and allows entry of a single integer number.

Paragraph - Displays and allows entry of multiple lines of text.

Text - Displays and allows entry of a single line of text.

Selection

Checkbox By Index - Displays a limited set of choices from which the user may select none, one, or many items and saves the indices of the selected choices.

Checkbox - Displays a limited set of choices from which the user may select none, one, or many items and saves the values of the selected choices.

Dropdown By Index - Displays a limited set of exclusive choices from which the user must select one item and saves the index of the selected choice.

Dropdown - Displays a limited set of exclusive choices from which the user must select one item and saves a value based on the selected choice.

Multiple Dropdown By Index - Displays a long list of choices from which the user may select none, one, or many items and saves the indices of the selected choices.

Multiple Dropdown - Displays a long list of choices from which the user may select none, one, or many items and saves values based on the selected choices.

Radio Button By Index - Displays a limited set of choices from which the user must select one item and saves a value based on the selected choice.

Radio Button - Displays a limited set of choices from which the user must select one item and saves a value based on the selected choice.

Display

Rich Text Types

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Bulleted List - Displays a bulleted list within a rich text component.

Header Text - Displays heading-styled text within a rich text component.

Inline Image - Displays an image within a rich text component.

List Item - Displays a numbered list within a rich text component.

Numbered List - Displays a numbered list within a rich text component.

Styled Icon - Displays a style icon within a rich text component.

Styled Text - Displays styled text within a rich text component.

Image Types

Document Image - Displays an image from document management.

User Image - Displays a user's profile photo.

Web Image - Displays an image from the web.

Document Viewer - Displays a document from document management on an interface.

Image - Displays an image from document management or the web.

Milestone - Displays the completed, current, and future steps of a process or sequence, such as a user's current step in a wizard or the current state of a business process.

Progress Bar - Displays a completion percentage, such as receiving all the necessary approval tasks, completing a certain number of on-boarding processes, or completing a single process.

Rich Text - Displays text in variety of styles, including bold, italics, underline, links, headers, and numbered and bulleted lists.

Time Display - Displays a single time (hour, minute, second). Does not take an input.

Video - Displays a video from the web

Web Content Field - Displays content from an external source.

Web Video - Displays a video from the web for use in a video field.

Action

Button Array Layout - Displays a list of buttons in the order they are specified.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Button Layout - Displays a list of buttons grouped by prominence. Use this layout in cases where prominence needs to be explicitly specified.

Link - Display one or more links of any type. Link types serve two main purposes: to navigate to objects, such as tasks and records, or to perform actions, such as downloading documents or starting processes.

Button Types

Button - Displays a button that can conditionally be used to submit a form.

Submit Button - Displays a button that can conditionally be used to submit a form.

Link Types

Authorization Link - Defines a link to authorize a user for a connected system that uses OAuth 2.0.

Document Download Link - Defines a link used to download a document.

Dynamic Link - Defines a link that triggers updates to one or more variables.

News Entry Link - Defines a link to news entries.

Process Task Link - Defines a link to a process task.

Record Link - Defines a link to a record view.

Report Link - Defines a link to a report.

Start Process Link - Defines a link to start a process and navigates the user through any initial chained forms.

Submit Link - Defines a link to trigger form submission.

User Record Link - Defines a link to a user record.

Web Link - Defines a link to an external web page.

Grids

Editable Grid - Arranges interface components in a tabular layout to provide quick inline editing of fields.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Paging Grid - Displays read-only text, links, and images in a grid that supports selecting, sorting, and paging.

Editable Grid Elements

Editable Grid Column Configuration - Defines a column configuration for use in an Editable Grid.

Editable Grid Header - Defines a column header for use in an Editable Grid.

Editable Grid Row - Displays a row of components within an editable grid.

Paging Grid Elements

Paging Grid Image Column - Displays a column of images within a paging grid.

Paging Grid Selection - Creates a selection configuration for use with a paging grid.

Paging Grid Text Column - Displays a column of text within a paging grid.

Charts

Bar Chart - Displays numerical data as horizontal bars. Use a bar chart to display several values at the same point in time.

Column Chart - Displays numerical data as vertical bars. Use a column chart to graphically display data that changes over time.

Line Chart - Displays a series of numerical data as points connected by lines. Use a line chart to visualize trends of data that changes over time.

Pie Chart - Represents numerical data as slices of a single circle. Use a pie chart to graphically display parts of a whole.

Chart Data

Chart Reference Line - Contains the reference line value for each threshold that is defined on a column, bar, or line chart.

Chart Series - Defines a series of data for a bar, column, line, or pie chart.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Pickers

Custom Picker - Displays an autocompleting input for the selection of one or more items from an arbitrary data set.

Document Picker - Displays an autocompleting input for the selection of one or more documents.

Document and Folder Picker - Displays an autocompleting input for the selection of one or more documents or folders.

Folder Picker - Displays an autocompleting input for selecting one or more folders.

Group Picker - Displays an autocompleting input for selecting one or more groups.

Record Picker - Displays an autocompleting input for the selection of one or more records, filtered by a single record type. Suggestions and picker tokens use the title of the record.

User Picker - Displays an autocompleting input for the selection of one or more users.

User and Group Picker - Displays an autocompleting input for selecting one or more users or groups.

Browsers

Document Browser - Displays the contents of a folder and allows users to navigate through a series of folders to find and download documents.

Document and Folder Browser - Displays the contents of a folder and allows users to navigate through a series of folders to find and select a folder or document.

Folder Browser - Displays the contents of a folder and allows users to navigate through a series of folders to find and select a folder.

Group Browser - Displays group membership structure in columns. Users can navigate through the structure and select a single user.

Hierarchy Browser (Columns) - Displays hierarchical data in the form of drillable columns with selectable cells.

Hierarchy Browser (Tree) - Displays hierarchical data in the form of a drillable tree.

Org Chart - Displays the organizational structure of users within Appian based on the users' Supervisor field values.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

User Browser - Displays group membership structure in columns. Users can navigate through the structure and select a single user.

User and Group Browser - Displays group membership structure in columns. Users can navigate through the structure and select a single user or group.

Hierarchy Browser Elements

Hierarchy Browser Node (Columns) - Returns a Hierarchy Browser Field Columns Node, used in the Node Configurations parameter of the Columns Browser to determine how items in the hierarchy are displayed.

Hierarchy Browser Node (Tree) - Returns a Tree Node, used in the Node Configurations parameter of the Tree Browser Component to determine how items in the hierarchy are displayed.

Interface Patterns

Interface patterns provide you with a combination of components and dynamic expressions to achieve common user-interface designs. This page contains a searchable list of all interface patterns.

Some patterns are available to drag-and-drop from the component palette in **DESIGN MODE**:

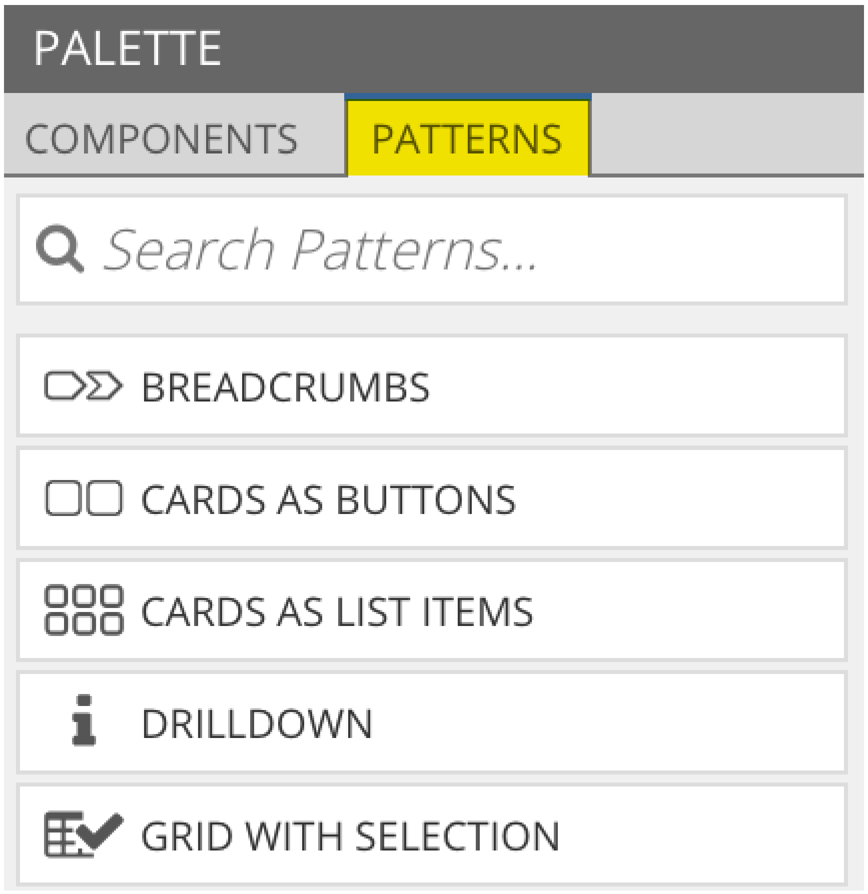
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Other patterns are available as an instructional recipe right here in the docs. Recipes contain an expression that you can copy and paste into your interface in **EXPRESSION MODE**.

Whether the pattern you're interested in is available in the **palette**, or from a **recipe**, be sure to check out How to Adapt a Pattern for Your Application if you're new at working with interfaces.

Many of the instructional recipes use the same employee data structure; the Use the Write to Data Store Entity Smart Service Function on an Interfacerecipe contains all of the employee data and objects necessary for these recipes to display property. If you have not already done so, and want employee data to see how other recipe's charts and grids work, implement that recipe first.

Search Patterns

All

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Document Management	Add Multiple File Upload Components Dynamically
Looping	Add Multiple Text Components Dynamically
Validation	Add Multiple Validation Rules to One Component
Validation, Grids, Looping	Add Validations to an Inline Editable Grid
Looping	Add and Populate Sections Dynamically
Grids, Looping	Add, Edit, and Remove Data in an Inline Editable Grid
Hierarchical Data, Group Management	Add, Remove, and Move Group Members Browser
Reports, Charts, Query Data, Grids	Aggregate Data from a Data Store Entity and Conditionally Display in a Chart or Grid
Reports, Charts, Query Data	Aggregate Data from a Data Store Entity and Display in a Chart
Reports, Charts, Query Data	Aggregate Data from a Data Store Entity by Multiple Fields and Display in a Chart
Reports, Charts, Query Data, Grids	Aggregate Data from a Data Store Entity on a Date or Date and Time Field
Reports, Charts, Query Data, Filtering	Aggregate Data from a Data Store Entity using a Filter and Display in a Chart
Formatting	Breadcrumbs
Hierarchical Data	Browse Hierarchical Data
Wizards	Build a Wizard with Milestone Navigation
Wizards	Build an Interface Wizard
Choice Components, Formatting, Conditional Display	Cards as Buttons
Choice Components, Images	Cards as List Items
Grids, Conditional Display	Conditionally Hide a Column in a Grid

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Validation	Configure Buttons with Conditional Requiredness
Conditional Display, Choice Components	Configure Cascading Dropdowns
Choice Components	Configure a Boolean Checkbox
Charts, Grids, Query Data	Configure a Chart Drilldown to a Grid
Charts, Grids	Configure a Chart to Grid Toggle
Choice Components	Configure a Dropdown Field to Save a CDT
Choice Components	Configure a Dropdown with an Extra Option for Other
Pickers	Configure an Array Picker
Pickers	Configure an Array Picker that Ignores Duplicates
Pickers	Configure an Array Picker with a Show All Option
Formatting	Define a Simple Currency Component
Grids	Delete Rows in a Grid
Grids, Formatting	Display Array of Data in a Grid
Grids, Records	Display Data with CDT Fields from in a Grid
Images, Grids	Display Images in a Grid
Document Management, Grids	Display Multiple Files in a Grid
Grids, Images, Reports	Display Processes by Process Model with Status Icons
Grids, Reports, Looping	Display a User's Tasks in a Grid With Task Links
Hierarchical Data, Grids	Expand/Collapse Rows in a Tree Grid

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Grids, Filtering	Filter the Data in a Grid
Charts, Grids, Filtering	Filter the Data in a Grid Using a Chart
Formatting, Data, Grids	Format Data in a Grid
Formatting	Format the User's Input
Grids	Grid with Selection Pattern
Formatting	KPIs
Validation, Grids	Limit the Number of Rows in a Grid That Can Be Selected
Validation	Make a Component Required Based on a User Selection
Grids, Smart Services, Filtering, Reports	Save a User's Report Filters to a Data Store Entity
Grids, Filtering, Reports	Searching on Multiple Fields
Conditional Display	Set a Numeric Rating Using Rich Text Icons
Default Value	Set the Default Value Based on a User Input
Default Value	Set the Default Value of CDT Fields Based on a User Input
Default Value	Set the Default Value of an Input on a Start Form
Default Value	Set the Default Value of an Input on a Task Form
Formatting, Grids	Show Calculated Columns in a Grid
Conditional Display	Show a Numeric Rating as Rich Text Icons
Validation	Showing Validation Errors that Aren't Specific to One Component
Formatting, Choice Components	Tabs

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Grids	Track Adds and Deletes in Inline Editable Grid
Records, Smart Services	Update an Entity-Backed Record from its Summary View
Grids	Use Links in a Grid to Show More Details About an Object
Grids	Use Links in a Grid to Show More Details and Edit Data
Grids, Web Services	Use Links in a Grid to Show More Details and Edit Data in External System
Grids	Use Selection For Bulk Actions in an Inline Editable Grid
Validation	Use Validation Groups for Buttons with Multiple Validation Rules
Filtering, Charts	Use a Filter to Adjust Chart Reference Lines
Smart Services, Grids, Looping	Use the Write to Data Store Entity Smart Service Function on an Interface
Looping	User List

Query Recipes

Overview

The recipes on this page show how to perform common data lookups using the `a!queryEntity()` expression function.

Although the examples on this page all deal with `a!queryEntity()`, the same patterns also apply to the `queryrecord()` function, which can be used to perform data lookups against entity-backed and process-backed records.

See also:

- Create Entity-Backed Records
- a!queryEntity()
- queryrecord()

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

The recipes can be worked on in no particular order. However, make sure to read the first section to get yourself set up.

Setup

Before we start with the recipes, we'll need a data store entity. For our examples, let's use the employee entity from the Records Tutorials.

See also: Records Tutorial

Next, create a constant called `EMPLOYEE_ENTITY` with Data Store Entity as the Type and the Employee entity as the Value.

See also: Constants

Retrieve the Data for All Fields

Goal: Retrieve the data for a all fields of an entity.

When you execute a query rule, it pulls back the data for all of the fields of the entity. This recipe replicates that functionality using `a!queryEntity()`. See Querying Data From an RDBMS on how to decide which function to use.

To retrieve all fields of the entity being queried, simply omit both the `selection` and `aggregation` parameters from `a!query()`. When using this approach, you should cast the result to the appropriate type to ensure that it works smoothly in process models and rules that use it. This is because `a!queryEntity()` always returns a datasubset that includes a dictionary, while query rules return a list of CDTs.

In this example we are going to retrieve the employee whose id is `8`.

Expression

```
cast(
  type!Employee,
  a!queryEntity(
    entity: cons!EMPLOYEE_ENTITY,
    query: a!query(
      filter: a!queryFilter(
        field: "id",
        operator: "=",
        value: 8
      ),
      pagingInfo: a!pagingInfo(
        startIndex: 1,
        batchSize: 1
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
    )
  )
 ).data
)

```

This example should return the value `[id=8, firstName=Jessica, lastName=Peterson, department=Finance, title=Analyst, phoneNumber=555-987-6543, startDate=2004-11-01]`. This value will be of type `Employee`.

If you expect your query to return multiple results, you should instead cast to a list of CDT. In this example, we will retrieve any employees whose first name begins with "A".

Expression

```

cast(
  typeof({type!Employee()}),
  a!queryEntity(
    entity: cons!EMPLOYEE_ENTITY,
    query: a!query(
      filter: a!queryFilter(
        field: "firstName",
        operator: "starts with",
        value: "A"
      ),
      pagingInfo: a!pagingInfo(
        startIndex: 1,
        batchSize: 5
      )
    )
  ).data
)
```

This example should return the value `[id=17, firstName=Andrew, lastName=Nelson, department=Professional Services, title=Consultant, phoneNumber=555-789-4560, startDate=3/15/2005]; [id=4, firstName=Angela, lastName=Cooper, department=Sales, title=Manager, phoneNumber=555-123-4567, startDate=10/15/2005]`.

Retrieve the Data for a Single Field

Goal: Retrieve the data for a single field of an entity rather than all of the fields.

When you execute a query rule it pulls back the data for all of the fields of the entity. The more data you pull back from the database the longer the query rule takes to run. A common way to restrict the amount of data returned by a query rule is to create several different data store entities that reference the same database table, each of which only contains some of the fields. Instead, using `a!queryEntity()` to select specific fields as shown below restricts the amount of returned data, is faster to develop, and has the advantage that the field or fields can be selected at run-time rather than design-time.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

In this example we are going to retrieve the phone number, stored in the field `phoneNumber`, for the employee whose id is `8`.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    selection: a!querySelection(
      columns: {
        a!queryColumn(
          field: "phoneNumber"
        )
      }
    ),
    filter: a!queryFilter(
      field: "id",
      operator: "=",
      value: 8
    ),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: 1
    )
  )
).data.phoneNumber[1]
```

This example should return the value `555-987-6543`.

To retrieve data for more than one field, you can add additional `a!queryColumn()`'s to the `columns` array.

Get the Distinct Values of a Field

Goal: Retrieve the unique list of values in a given field.

It will almost always be significantly faster to have the data source do the uniqueness calculation before returning the data to Appian. This is especially true for large data sets. `a!queryEntity()` lets the data source perform the uniqueness calculation.

In this example we are going to retrieve the list of departments that have employees.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    aggregation: a!queryAggregation(
      aggregationColumns: {
        a!queryAggregationColumn(
          field: "department",
          isGrouping: true
        )
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
      }
    ),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: -1,
      sort: a!sortInfo(
        field: "department",
        ascending: true
      )
    )
  )
).data.department

```

This example should return a list containing "Engineering", "Finance", "HR", "Professional Services", and "Sales". Note that even though there is more than one employee in many of these departments, each department is only listed once in the result.

Aggregating on a Field

Goal: Perform an aggregation or computation on all values of field.

In this example we are going to count the number of employees in each department.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    aggregation: a!queryAggregation(
      aggregationColumns: {
        a!queryAggregationColumn(
          field: "department",
          isGrouping: true
        ),
        a!queryAggregationColumn(
          field: "department",
          alias: "numberOfEmployees",
          aggregationFunction: "COUNT"
        )
      }
    ),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: -1,
      sort: a!sortInfo(
        field: "department",
        ascending: true
      )
    )
  )
)
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

This example should return one dictionary for each department where the keys in the dictionary are `department`and `numberOfEmployees` and the values match the following table.

department	**numberOfEmployees**
Engineering	6
Finance	4
HR	2
Professional Services	4
Sales	4

Aggregating on Year and Month

Goal: Perform a time aggregation on all values of field.

In this example we are going to count the number of employees that started on a specific month and year.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    aggregation: a!queryAggregation(
      aggregationColumns: {
        a!queryAggregationColumn(
          field: "startDate",
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
        alias: "year_startDate",
        isGrouping: true,
        groupingFunction: "YEAR"
      ),
      a!queryAggregationColumn(
        field: "startDate",
        alias: "month_startDate",
        isGrouping: true,
        groupingFunction: "MONTH"
      ),
      a!queryAggregationColumn(
        field: "id",
        alias: "idCount",
        aggregationFunction:"COUNT"
      )
    }
  ),
  pagingInfo: a!pagingInfo(
    startIndex: 1,
    batchSize: -1
  )
),
fetchTotalCount: true
)

```

This example should return a list of dictionaries for each distinct year and month combination with the count of the employees that have a start date in that month and year.

Querying on Multiple Conditions

Goal: Retrieve data that meets at least one of two different conditions.

Using query rules, the only way to find entries that match at least one of two conditions is to run two different query rules and combine the results. Using a `logicalExpression` inside the Query object we can execute the same logic in a single call to the data source, resulting in faster performance.

In this example we are going to retrieve the names of employees who **either** started within the last 2 years **or**have the word "Associate" in their title.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    selection: a!querySelection(
      columns: {
        a!queryColumn(field: "firstName"),
        a!queryColumn(field: "lastName")
      }
    ),
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
    logicalExpression: a!queryLogicalExpression(
      operator: "OR",
      filters: {
        a!queryFilter(
          field: "startDate",
          operator: ">",
          value: date(year(now())-2, month(now()), day(now()))
        ),
        a!queryFilter(
          field: "title",
          operator: "includes",
          value: "Associate"
        )
      }
    ),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: -1
    )
  )
)

```

The exact list of results that is returned will to depend on when you run the example:

- Before `Jan 2, 2015`: `John Smith` will be the only employee returned because of the start date condition.
- On or after `Jan 2, 2015`: no employees will be included in the results because of their start date.

`Elizabeth Ward`, `Laura Bryant` and `Stephen Edwards` will be included in the result regardless of when you run the example as they are included because their title contains the word `Associate`

Querying on Nested Conditions

Goal: Retrieve data based on complex or nested conditions.

In this example we are going to retrieve the names of the senior members of the Engineering department where "senior" is defined as either having a title of "Director" or having a start date of more than 10 years ago.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    selection: a!querySelection(
      columns: {
        a!queryColumn(field: "firstName"),
        a!queryColumn(field: "lastName")
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
      }
    ),
    logicalExpression: a!queryLogicalExpression(
      operator: "AND",
      filters: a!queryFilter(
        field: "department",
        operator: "=",
        value: "Engineering"
      ),
      logicalExpressions: {
        a!queryLogicalExpression(
          operator: "OR",
          filters: {
            a!queryFilter(
              field: "startDate",
              operator: "<",
              value: date(year(now())-10, month(now()), day(now()))
            ),
            a!queryFilter(
              field: "title",
              operator: "includes",
              value: "Director"
            )
          }
        )
      }
    ),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: -1
    )
  )
)

```

This example should return `John Smith` and `Mary Reed`. `John Smith` is included because he is a Director and Mary Reed is included because her start date is more than 10 years ago. Both of them are in the Engineering department.

Filtering for Null Values

Goal: Find entries where a given field is null.

In this example we are going to find all employees who are missing either `firstName`, `lastName`, `department`, `title`, `phoneNumber`, or `startDate`.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    selection: a!querySelection(
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
      columns: {
        a!queryColumn(field: "firstName"),
        a!queryColumn(field: "lastName")
      }
    ),
    logicalExpression: a!queryLogicalExpression(
      operator: "OR",
      filters: {
        a!queryFilter(field: "firstName", operator: "is null"),
        a!queryFilter(field: "lastName", operator: "is null"),
        a!queryFilter(field: "department", operator: "is null"),
        a!queryFilter(field: "title", operator: "is null"),
        a!queryFilter(field: "phoneNumber", operator: "is null"),
        a!queryFilter(field: "startDate", operator: "is null")
      }
    ),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: -1
    )
  )
)

```

This example does not return any results because none of the employees in our sample data are missing any of the specified fields.

Searching on Multiple Fields

Goal: Retrieve data based on search criteria specified by end users e.g. when looking for employees by last name, title, or department. Search criteria that are left blank are not included in the query.

For an example on filtering for null values, see the recipe: Filtering for Null Values.

Expression

First, create an expression rule `ucSearchEmployees` with the following rule inputs:

- lastName (Text)
- title (Text)
- department (Text)
- pagingInfo (Any Type)

Enter the following definition for the rule:

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    logicalExpression: if(
      and(
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
        isnull(ri!lastName),
        isnull(ri!title),
        isnull(ri!department)
      ),
      null,
      a!queryLogicalExpression(
        operator: "AND",
        filters: {
           if(
            isnull(ri!lastName),
            {},
            a!queryFilter(field: "lastName", operator: "includes", value: ri!lastName)
          ),
          if(
            isnull(ri!title),
            {},
            a!queryFilter(field: "title", operator: "includes", value: ri!title)
          ),
          if(
            isnull(ri!department),
            {},
            a!queryFilter(field: "department", operator: "=", value: ri!department)
          )
        }
      )
    ),
    pagingInfo: ri!pagingInfo
  ),
  /* The fetchTotalCount parameter should be set to true when the totalCount retutned */
  /* in the datasubset is required. This example assumes the result will be used in a */
  /* grid where totalCount is required for paging. */
  fetchTotalCount: true
)

```

Test it out

Unlike the recipes above, this one is a rule with inputs. So rather than just getting a single result let's take a look at several different results for different rule inputs.

First, let's try not specifying any fields except for pagingInfo:

```

  rule!ucSearchEmployees(
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: 30,
      sort: a!sortInfo(
        field: "lastName",
        ascending: true
      )
    )
  )
```

This query will return the first 30 employees, sorted A-Z by last name.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

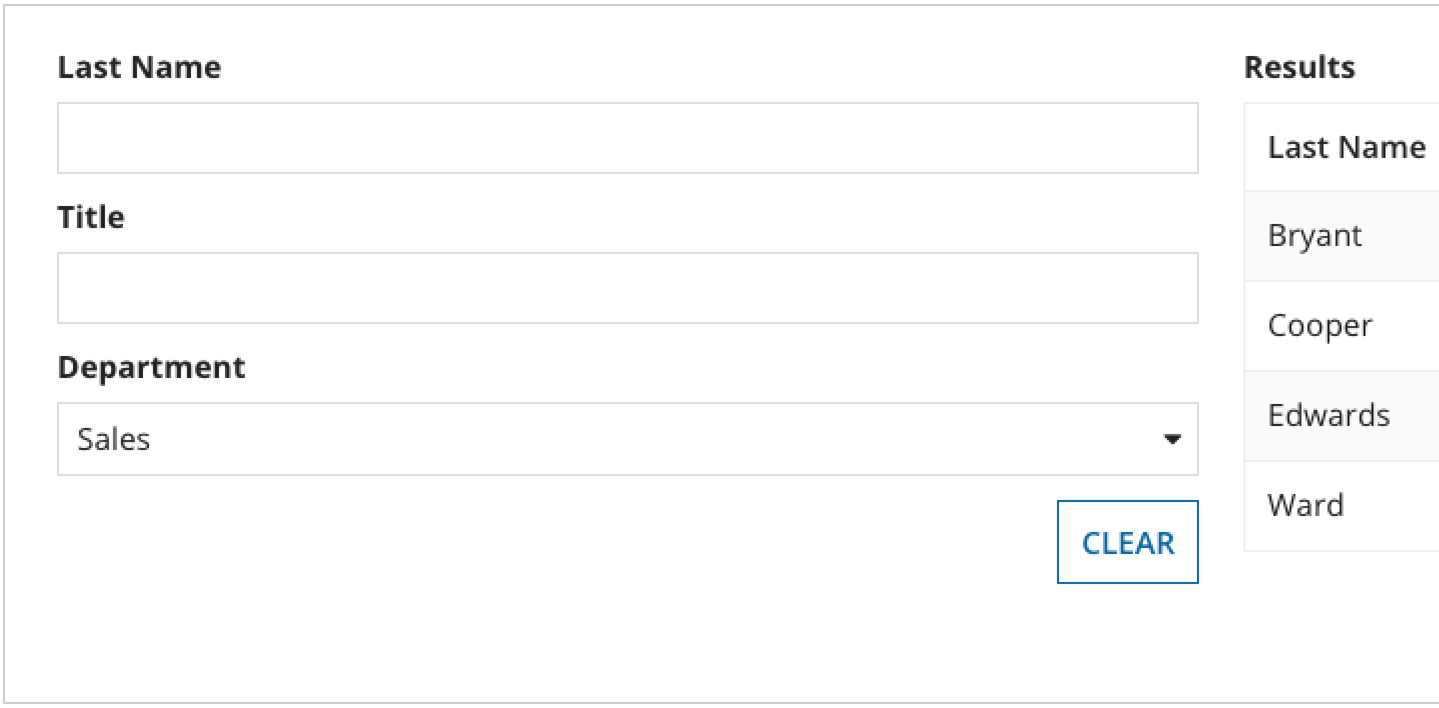
After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Next let's try specifying a department in addition to the pagingInfo:

```
rule!ucSearchEmployees(
  department: "Sales",
  pagingInfo: a!pagingInfo(
    startIndex: 1,
    batchSize: 30,
    sort: a!sortInfo(
      field: "lastName",
      ascending: true
    )
  )
)
```

This expression will return a list of employees in the Sales department, sorted A-Z by last name. In this example that is: `Angela Cooper`, `Laura Bryant`, `Stephen Edwards`, and `Elizabeth Ward`.



Last Name

Title

Department

Sales

Results

Last Name
Bryant
Cooper
Edwards
Ward

We can also combine multiple filters together. Let's try searching by both last name and department:

```
rule!ucSearchEmployees(
  lastName: "Bryant",
  department: "Sales",
  pagingInfo: a!pagingInfo(
    startIndex: 1,
    batchSize: 30,
    sort: a!sortInfo(
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
        field: "lastName",
        ascending: true
      )
    )
  )

```

This expression will return a list of employees that are in the `Sales` department and have a last name that contains `Bryant`. In this case that's a single employee: `Laura Bryant`.

To see an example of integrating this query into an interface, see the Interface Recipe: Searching on Multiple Fields

Sorting on Multiple Fields without Using Aggregations

Goal: Demonstrate how to sort on multiple columns when there is no field aggregation.

Using the *a!querySelection* function allows you to define a set of column selection configurations. When using this function, you can sort on any of the fields in the data entity, whether the fields are included in the selection or not.

In this example we are going to retrieve the *department*, *first name*, and *last name* of employees and sort them ascendingly by department, then by *title*. The data will be sorted first by department, and then by title, eventhough title is not part of the query selection.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    selection: a!querySelection(columns: {
      a!queryColumn(field: "department"),
      a!queryColumn(field: "firstName"),
      a!queryColumn(field: "lastName"),
    }),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: 20,
      sort: {
        a!sortInfo(
          field: "department",
          ascending: true
        ),
        a!sortInfo(
          field: "title",
          ascending: true
        )
      }
    )
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
  ),
  /* The fetchTotalCount parameter should be set to true when the totalCount retutned */
  /* in the datasubset is required. This example assumes the result will be used in a */
  /* grid where totalCount is required for paging. */
  fetchTotalCount: true
)
```

Sorting on Multiple Fields when Aggregation is Applied

Goal: Demonstrate how to sort on multiple fields when aggregation is performed in one or more fields.

In this example we are going to retrieve the count of *employees* by *department* and *title*. We are also going to sort the results ascendingly; first by department, then by title.

Expression

```
a!queryEntity(
  entity: cons!EMPLOYEE_ENTITY,
  query: a!query(
    aggregation: a!queryAggregation(aggregationColumns: {
      a!queryAggregationColumn(field: "department", isGrouping: true),
      a!queryAggregationColumn(field: "title", isGrouping: true),
      a!queryAggregationColumn(field: "lastName", aggregationFunction: "COUNT"),
    }),
    pagingInfo: a!pagingInfo(
      startIndex: 1,
      batchSize: 20,
      sort: {
        a!sortInfo(
          field: "department",
          ascending: true
        ),
        a!sortInfo(
          field: "title",
          ascending: true
        )
      }
    )
  )
)
```

When using *a!queryAggregation*, you can only sort on fields that are part of the query aggregation. Unlike when using *a!querySelection*, you must aggregate on a field if you want to sort by it.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Function Recipes

Overview

This page lists a collection of function recipes you can use throughout the Appian application. Similar to a cookbook, it shows the individual ingredients that go into creating an expression using Appian functions, the order in which to combine them, and tips on when to use them or modify for preference.

The expressions listed below are not the only way to accomplish each desired outcome. This page is intended to teach you methods for creating expressions using Appian functions that are practical and efficient, as well as provide working expressions you can implement as is. Just as with cooking recipes, you may want to alter the values in each recipe to accommodate your specific requirements.

The function recipes are categorized by the type of output. Each recipe is titled with the question in mind, "What do you want to do?"

NOTE: For function recipes that require a rule as an Ingredient, create the rule as listed before implementing the function recipe.

Date/Time Results

Retrieve Next Anniversary Date

Use Case

You want to retrieve next year's anniversary date based on a start date, such as an employee's hire date of **02/05/2011**.

Ingredients

- if()
- today()
- date()
- day()
- month()
- year()

Inputs

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

- start (date)

Expression

```
if(
  and(month(ri!start) <= month(today()),day(ri!start) <= day(today())),
  date(1 + year(today()), month(ri!start), day(ri!start)),
  date(year(today()), month(ri!start), day(ri!start))
)
```

Start a Timer One Minute After a Process Starts

Use Case

You want to set a timer to start exactly one minute after a process starts, as compared to automatically.

Ingredients

- pp!start_time
- minute()

Expression

Configure this in the "Delay until the date and time specified by this expression" on the Setup tab for the Timer Event.

```
pp!start_time + minute(1)
```

NOTE: To change the time difference from 1 minute to more, modify the value in the `minute()` function as desired.

Add Ten Minutes to a DateTime Value

Use Case

You want to add ten minutes to a datetime value saved as pv!datetime.

Ingredients

- datetime
- intervalds()

Expression

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
pv!datetime + intervalds(0,10,0)
```

NOTE: To change interval added to the datetime value, modify the values in `intervalds()` as desired.

Determine if a Date-Time Value Falls Within Working Hours

Use Case

Users are able to enter any date-time value, but you want to perform an extra validation after users submit the form to determine if a date-time value (saved as **ri!dt**) falls between your company's working hours of 6:00 AM and 5:00 PM.

The idea is to work with one consistent time scale, which is done here by the use of timestamps rather than both hours and minutes. This function constructs new timestamps for the boundaries matching the input.

Ingredients

- or()
- gmt()
- datetime()
- year()
- month()
- day()

Inputs

- dt (DateTime)

Expression

```
or(
  ri!dt > gmt(
    datetime(
      year(ri!dt),
      month(ri!dt),
      day(ri!dt),
      17,
      0,
      0
    )
  ),
  ri!dt < gmt(
    datetime(
      year(ri!dt),
      month(ri!dt),
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
      day(ri!dt),
      6,
      0,
      0
    )
  )
)
```

NOTE: To change the working hours, modify the values within the `datetime()` function as desired

Display the Number of Days Before a Specific Date

Use Case

You want to tell users how many business days are left before a deal closes, if the deal has already closed, or if the deal is closing today.

Ingredients

- if()
- networkdays()
- today()

Inputs

- closeDate (Date)

Expression

```
if(
  networkdays(today(), ri!closeDate)<=0,
  "This deal has closed.",
  if(
    networkdays(today(), ri!closeDate)=1,
    "This deal will close at the end of today.",
    "This deal will close in" & networkdays(today(), ri!closeDate) & "business days."
  )
)
```

NOTE: To change the text that displays, modify the text as desired. To base the number of days on a system calendar, replace any instance of `networkdays(today(), ri!closeDate)` with `calworkdays(today(), ri!closeDate, "companyCalendar")` where `companyCalendar` is the name of your system calendar. To include all days (including weekends), replace any instance of `networkdays(today(), ri!closeDate)` with `tointeger(ri!closeDate - today())`.

See also: calworkdays() and tointeger()

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Convert an XML String into a Value of Type Datetime

Use Case

You have a localized XML string representing a date and time and need to convert it to a value of type Datetime in the GMT timezone.

Ingredients

- with()
- split()
- left()
- right()
- datetime()
- gmt()

Inputs

- dateText (Text)

Expression

```
=with(
  local!fullArray: split(ri!dateText, "T"),
  local!dateArray: split(local!fullArray[1], "-"),
  local!timeArray: split(left(local!fullArray[2], 12), ":"),
  local!smsArray: split(local!timeArray[3], "."),
  local!timeZone: "GMT" & right(local!fullArray[2], 6),
  local!rawDate: datetime(
    local!dateArray[1],
    local!dateArray[2],
    local!dateArray[3],
    local!timeArray[1],
    local!timeArray[2],
    local!smsArray[1],
    local!smsArray[2]
  ),
  gmt(
    local!rawDate,
    local!timeZone
 )
)
```

yields `9/25/2013 1:50 AM GMT+00:00` where `ri!dateText = 2013-09-24T19:50:24.192-06:00`

Number Results

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Return the Number of Active Employees in Your Application

Use Case

You want to quickly display how many employee records exist in your application.

Ingredients

- topaginginfo()
- totalCount

Expression

- `myQueryRule(topaginginfo(1,0)).totalCount`

NOTE: When using a batchsize of 0 and attempting to only retrieve a total count based on a query, the function detailed above is better to use than `count(myQueryRule())`. If you used `count(myQueryRule())`, the system would run the main query; whereas using `.totalcount`, the system only executes a count query against the database.

Return the Total Number of Revisions Made to a Document

Use Case

You want to show the number of times a document has had a new version saved.

Ingredients

- document()

Input

- doc (Document)

Expression

- `document(ri!doc,"totalNumberOfVersions")-1)`

Return a Random Integer in a Range

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Use Case

You want to return a random integer between two integers, inclusive.

Ingredients

- rand()
- tointeger()

Inputs

- min (Number (Integer))
- max (Number (Integer))

Expression

```
ri!min + tointeger(rand() * (ri!max - ri!min))
```

Text Results

Truncate Text After 50 Characters

Use Case

You want to truncate the remaining part of a text value once it surpasses 50 characters by replacing the excess text with an ellipsis (…).

Ingredients

- if()
- len()

Inputs

- text (Text)

Expression

```
if(
  len(ri!text) > 50,
  left(ri!text, 50) & "...",
  ri!text
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
)
```

NOTE: To change the amount of characters allowed before truncating, modify the numeric value as desired.

Display the Full Name of a User

Use Case

You want to display the full name of a user despite knowing only the username. Additionally, the field that is storing user information may not always contain a value.

Ingredients

- if()
- isnull()
- user()

Inputs

- user (User)

Expression

```
if(
  isnull(ri!user),
  "",
  user(ri!user, "firstName") & " " & user(ri!user, "lastName")
)
```

NOTE: To change the user information that is populated, modify the second parameter in the `user()` function as desired.

Create a Title from any Text Value

Use Case

You want to convert a text value to display as a title, which will capitalize the first word, and all words not found in a "no capitalize" list.

Ingredients

- proper()
- split()
- trim()

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

- with()
- a!forEach()
- if()
- contains()
- and()
- not()

Inputs

- title (Text)

Expression

```
with(
  /* Breaks up the title into an array of words and removes whitespaces. */
  local!titleArray: split(trim(ri!title), " "),
  /* A list of words that will be ignored. Adjust this list to change what does not get capitalized. */
  local!noCaps:
{"a","an","the","at","by","in","of","up","as","and","but","or","for","nor","etc.","etc","on","at","to","from", "that"},
  /* If the word is in local!noCaps but not the first word, don't capitalize. Otherwise capitalize the word. */
  concat(
    a!forEach(
      items: local!titleArray,
      expression: if(
        and(not(fv!isFirst), contains(local!noCaps, fv!item)),
        fv!item,
        proper(fv!item)
      ) & if(fv!isLast, "", " ")
    )
  )
)
```

NOTE: To produce initial caps for your own text, modify the text value as desired.

Array Results

Remove all Values in an Array

Use Case

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

You want to remove all values in an array so it is considered empty.

Ingredients

- ldrop()
- count()

Inputs

- array (Any Array)

Expression

```
ldrop(ri!array, count(ri!array))
```

NOTE: Instead of the word array, enter the name or array reference as needed.

Repeat Each Item in an Array Based on the Values of a Different Array

Use Case

You want to repeat each item in an array (for example, `local!textList`) a certain number of times as defined by the values in a different array (for example, `local!frequencyList`).

- Where `local!textList = {"a", "b", "c"}` and `local!frequencyList = {2, 3, 1}`, the output would be `{"a", "a", "b", "b", "b", "c"}`.

Ingredients

- a!forEach()
- repeat()

Inputs

- textList (Array)
- frequencyList (Array)

Expression

```
load(
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
  local!textList: {"a","b","c"},
  local!frequencyList:{2,3,1},
  a!forEach(
    items:local!textList,
    expression: repeat(
      local!frequencyList[fv!index],
      fv!item
    )
  )
)
```

NOTE: To change which variables to use, modify the textList value to the array variable that should determine the text to repeat and modify the frequencyList value to the array variable that should determine how many times each text value in the aforementioned array should be repeated.

Extract a List from an EntityData Array of the Data Written to a Single Entity

Use Case

You want to extract the list of data written to an entity (for example, Opportunity) based on the output generated by using the Write to Multiple Data Store Entities Smart Service to write to three entities: Opportunity, Contact, and Line Item.

See also: Write to Multiple Data Store Entities Smart Service

The explanation of the configuration is longer than the solution. If you configure the Write to Multiple Data Store Entities Smart Service to update data for three entities (Opportunity, Contact, and Line Item), each entity may show up more than once in the EntityData input array. Consequently, the output of the smart service ("Stored Values") would contain a dynamic number of elements for each EntityData.

To get the list of all Opportunities that were updated by the smart service, you need to append every Data field where the entity is equal to Opportunity.

Ingredients

- a!forEach()
- Write to Multiple Data Store Entity Smart Service
    - StoredValue (EntityData)
- OPPORTUNITY_ENTITY Constant (Data Store Entity)
    - `entityData` is the array to operate on and `entity` is the context parameter

Expression

Within the expression editor of a new custom output:

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
a!forEach(
  items: merge(ac!StoredValues, cons!OPPORTUNITY_ENTITY),
  expression: if(
    fv!item[1].entity = fv!item[2],
    fv!item.Data,
    {}
  )
)
```

NOTE: To change the entity by which to pull the list of data from, modify the OPPORTUNITY_ENTITY value as desired.

Sort an Array

Use Case

You want to sort an array of values.

Ingredients

- todatasubset()
- a!forEach()
- a!pagingInfo()
- a!sortInfo()

Inputs

- array (Any Array)

Expression

```
todatasubset(
  a!forEach(
    ri!array,
    {
       value: fv!item
    }
  ),
  a!pagingInfo(
    startIndex: 1,
    batchSize: -1,
    sort: a!sortInfo(
      field:"value",
      ascending: true
    )
  )
).data.value
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Boolean Results

Determine if Items in an Array are Contained Within Another Array

Use Case

You want to see if any of the items in an array are contained within another array.

Ingredients

- contains()
- a!forEach()
- or()

Expression

```
or(
  a!forEach(
    items:ri!originalArray,
    expression: contains(ri!compareArray, fv!item)
  )
)
```

Matching Results

Return a CDT from an Array that Contains a Field Value Matching Another Value

Use Case

You have an array of department CDTs, each containing a field called `Id`, and you want to retrieve the CDT with an `id` value matching the value of a process variable.

Ingredients

- displayvalue()
- rule input: `ri!departmentid` (Integer)
- rule input: `ri!departments` (CDT Array)
- Custom Data Type:
  ```
      department (Text)
          |- id (Integer)
  ```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
3           |- address (Text)
```

Expression

```
displayvalue(ri!departmentId, ri!departments.id, ri!departments, "none")
```

NOTE: To change the value that displays if the `ri!departmentId` doesn't match any `ri!department.id` values, modify the "none" text value as desired.

Process Nodes and Smart Services

The Appian process modeler pallette contains all of the nodes and Appian smart services that can be used to define a process workflow. These activities are broken into two main categories: standard nodes and smart services. Standard nodes consists of standard BPMN activites, events, and gateways. Smart services are flow activities that integrate specialized business services, like sending e-mails or writing data to a database. Smart services consists of Appian Smart Services and Integration Services.

Refer to the following tables for more information about specific nodes or smart services:

Standard Nodes

Smart Services

Standard Nodes

Standard nodes consist of Activities, Events, and Gateways. Activities are used within process workflow to capture or process business data. Events allow designers to start, stop, or continue the progress of workflows. Gateway are used for workflow control.

Icon	Name	Description
	Script Task	The Script Task is used to perform an automated activity. Script Tasks are often used for data queries & transformation between other activities in a workflow

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Description
	Sub-Process	The Sub-Process Activity is used to launch child processes from within your parent process, allowing for data transfer between them. Sub-Processes can be run either synchronously or asynchronously.
☑	User Input Task	The User Input Task activity is used to assign a task to a user and/or group. InterfacesInterfaces.md are associated within a User Input Task to capture data from a form into a process.
○	End Event	An End Event is used to denote the end of a process flow within a process model. The process remains active until all active paths in the process arrive at an end event.
✉	Receive Message Event	The Receive Message event can consume an action, in the form of a Java message or e-mail. When a receive message event is configured as a process node on the designer canvas, the event is activated when the process flow reaches the event. The process flow only proceeds once the event has completed execution.
☰	Rule Event	Rule events can either be added to the process flow. They are used when there are certain conditions that need to be met before the process flow proceeds. Once an intermediate rule event is activated, it remains active until the rule evaluates to true.
✉	Send Message Event	In a process, a Send Message event can be used to generate a message that is made available to all Receive Message events (which are actively listening for messages). Messages sent to events that are not active are discarded.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Description
○	Start Event	A Start Event node denotes the beginning of a process in a process model, and is used to configure how a process is launched.
◷	Timer Event	Timer events can added to the process flow to schedule a process activity, or to only continue the flow when certain conditions are met.
✚	AND Gateway	The AND Gateway directs all incoming workflow(s) to all of the possible branches. If more than one incoming path is used, all incoming paths must reach the node before the process can continue.
✳	Complex Gateway	This type of gateway allows you to selectively accept (or restrict) incoming paths and evaluate rules to determine outgoing paths. For example, you can restrict the node to accept only the first three out of four incoming paths, or require input from certain nodes before continuing. Outgoing paths can be configured in the same manner as other gateway nodes.
◯	OR Gateway	An OR Gateway directs incoming flows to one of many possible output paths, based on the condition(s) you set.
✕	XOR Gateway	The exclusive (XOR) gateway connects one incoming path with a single outgoing path. The outgoing path is chosen from a number of possible paths and determined by one or more conditions that you set.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Smart Services

Smart services provide specialized business services. The two categories of smart services are Appian Smart Services and Integration Services. Smart services are, by default, unattended, meaning the activity will execute once activated. However, certain smart services can be configured as attended. Many of the attended smart services also have an associated smart service function available, which can be used in an Appian expression to invoke that smart service independent of a process model.

Icon	Name	Function Name (for use in Appian expressions)
	Execute Process Report Smart Service	Not available for this smart service.
	Send Email Smart Service	Not available for this smart service.
	Export Data Store Entity to CSV Smart Service	a!exportDataStoreEntityToCsv()
	Export Data Store Entity to Excel Smart Service	a!exportDataStoreEntityToExcel()
	Export Process Report to CSV Smart Service	a!exportProcessReportToCsv()
	Export Process Report to Excel Smart Service	a!exportProcessReportToExcel()

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	HTML Doc From Template Smart Service	Not available for this smart service.
	Open Office Writer Doc From Template Smart Service	Not available for this smart service.
	PDF Doc From Template Smart Service	Not available for this smart service.
	Text Doc From Template Smart Service	Not available for this smart service.
	Word Doc from Template Smart Service	Not available for this smart service.
	Create Folder Smart Service	a!createFolder()
	Create Knowledge Center Smart Service	a!createKnowledgeCenter()
	Create Knowledge Center Smart Service (17.4)	a!createKnowledgeCenter_17r4()
	Delete Document Smart Service	a!deleteDocument()
	Delete Folder Smart Service	a!deleteFolder()

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Delete KC Smart Service	a!deleteKnowledgeCenter()
	Edit Document Properties Smart Service	a!editDocumentProperties()
	Edit KC Properties Smart Service	a!editKnowledgeCenterProperties()
	Lock Document Smart Service	a!lockDocument()
	Modify Folder Security Smart Service	a!modifyFolderSecurity()
	Modify KC Security Smart Service	a!modifyKnowledgeCenterSecurity()
	Move Document Smart Service	a!moveDocument()
	Move Folder Smart Service	a!moveFolder()
	Rename Folder Smart Service	a!editFolderProperties()
	Unlock Document Smart Service	a!unlockDocument()

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Add Group Admins Smart Service	a!addAdminsToGroup()
	Add Group Members Smart Service	a!addMembersToGroup()
	Add User Smart Service	a!createUser()
	Change User Type Smart Service	a!updateUserType()
	Create Group Smart Service	a!createGroup()
	Deactivate User Smart Service	a!deactivateUser()
	Delete Group Smart Service	a!deleteGroup()
	Edit Group Smart Service	a!editGroup()
	Join Group Smart Service	Not available for this smart service.
	Leave Group Smart Service	Not available for this smart service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Modify User Security Smart Service	a!modifyUserSecurity()
	Reactivate User Smart Service	a!reactivateUser()
	Remove Group Admins Smart Service	a!removeGroupAdmins()
	Remove Group Members Smart Service	a!removeGroupMembers()
	Set Group Attributes Smart Service	a!setGroupAttributes()
	Update User Profile Smart Service	a!updateUserProfile()
	Cancel Process Smart Service	a!cancelProcess()
	Cancel Process Smart Service (17.3)	a!cancelProcess_17r3()
	Complete Task Smart Service	a!completeTask()
	Modify Process Security Smart Service	Not available for this smart service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Start Process Smart Service	a!startProcess()
	Increment Constant Smart Service	Not available for this smart service.
	Start Rule Tests (All)	a!startRuleTestsAll()
	Start Rule Tests (Applications)	a!startRuleTestsApplications()
	Update Constant Smart Service	Not available for this smart service.
	Follow Records Smart Service	Not available for this smart service.
	Follow Users Smart Service	Not available for this smart service.
	Post Comment to Feed Entry Smart Service	Not available for this smart service.
	Post Event to Feed Smart Service	Not available for this smart service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Post Hazard to Feed Entry Smart Service	Not available for this smart service.
	Post System Event to Feed Smart Service	Not available for this smart service.
	Call Integration Smart Service	Not available for this smart service.
	Call Web Service Smart Service	Not available for this smart service.
	HTTP File Download Smart Service	Not available for this smart service.
	HTTP File Upload Smart Service	Not available for this smart service.
	Invoke SAP BAPI Smart Service	a!sapInvokeWithCommit()
	Query Database Smart Service	Not available for this smart service.
	Delete from Data Store Entities Smart Service	a!deleteFromDataStoreEntities()

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Write to Data Store Entity Smart Service	a!writeToDataStoreEntity()
	Write to Multiple Data Store Entities Smart Service	a!writeToMultipleDataStoreEntities()
	Add Attachment Smart Service [Deprecated]	Not available for this smart service.
	Create Case Management Page Smart Service [Deprecated]	Not available for this smart service.
	Create Community Smart Service [Deprecated]	Not available for this smart service.
	Create Department Smart Service [Deprecated]	Not available for this smart service.
	Create Forum Smart Service [Deprecated]	Not available for this smart service.
	Create New Version Smart Service [Deprecated]	Not available for this smart service.
	Create Page Smart Service [Deprecated]	Not available for this smart service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Create Team Smart Service [Deprecated]	Not available for this smart service.
	Deactivate Community Properties Smart Service [Deprecated]	Not available for this smart service.
	Delete Community Smart Service [Deprecated]	Not available for this smart service.
	Delete Department Smart Service [Deprecated]	Not available for this smart service.
	Delete Forum Smart Service [Deprecated]	Not available for this smart service.
	Delete Message Smart Service [Deprecated]	Not available for this smart service.
	Delete Page Smart Service [Deprecated]	Not available for this smart service.
	Delete Team Smart Service [Deprecated]	Not available for this smart service.
	Delete Topic Smart Service [Deprecated]	Not available for this smart service.
	Edit Community Properties Smart Service [Deprecated]	Not available for this smart service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Edit Department Properties Smart Service [Deprecated]	Not available for this smart service.
	Edit Page Properties Smart Service [Deprecated]	Not available for this smart service.
	Edit Team Properties Smart Service [Deprecated]	Not available for this smart service.
	Export to Excel/CSV Smart Service [Deprecated]	Not available for this smart service.
	HTTP Query Smart Service [Deprecated]	Not available for this smart service.
	Modify Community Security Smart Service [Deprecated]	Not available for this smart service.
	Modify Forum Security Smart Service [Deprecated]	Not available for this smart service.
	Move Community Smart Service [Deprecated]	Not available for this smart service.
	Move KC Smart Service [Deprecated]	Not available for this smart service.
	Move Topic Smart Service [Deprecated]	Not available for this smart service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
	Page Security Smart Service [Deprecated]	Not available for this smart service.
	Post Message Smart Service [Deprecated]	Not available for this smart service.
	Publish Page Smart Service [Deprecated]	Not available for this smart service.
	Reactivate Community Properties Smart Service [Deprecated]	Not available for this smart service.
	Send Alert Smart Service [Deprecated]	Not available for this smart service.
	Set Global Home Page Smart Service [Deprecated]	Not available for this smart service.
	Set Group Home Page Smart Service [Deprecated]	Not available for this smart service.
	Set User Home Page Smart Service [Deprecated]	Not available for this smart service.
	Set User Storage Space Smart Service [Deprecated]	Not available for this smart service.
	Start Topic Smart Service [Deprecated]	Not available for this smart service.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Icon	Name	Function Name (for use in Appian expressions)
▢	Upload Document Smart Service [Deprecated]	Not available for this smart service.

Sites

Sites provide a customized user experience that is focused on a specific set of functionality. Designers can create sites with a business-oriented navigation experience and terminology their users are familiar with.

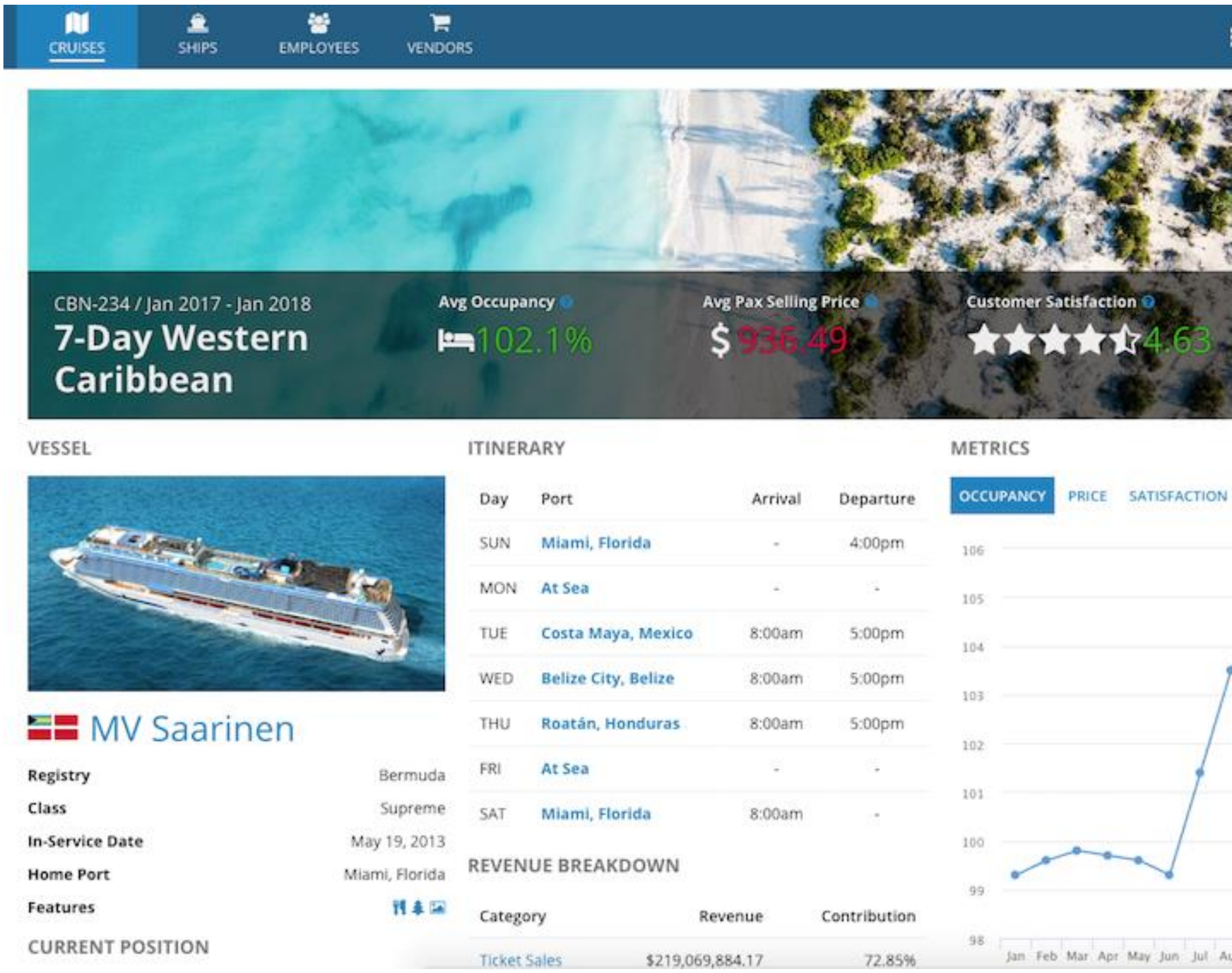
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



CBN-234 / Jan 2017 - Jan 2018

7-Day Western Caribbean

Avg Occupancy 102.1%

Avg Pax Selling Price $ 936.49

Customer Satisfaction 4.63

VESSEL

MV Saarinen

Registry	Bermuda
Class	Supreme
In-Service Date	May 19, 2013
Home Port	Miami, Florida
Features	

CURRENT POSITION

ITINERARY

Day	Port	Arrival	Departure
SUN	Miami, Florida	-	4:00pm
MON	At Sea	-	-
TUE	Costa Maya, Mexico	8:00am	5:00pm
WED	Belize City, Belize	8:00am	5:00pm
THU	Roatán, Honduras	8:00am	5:00pm
FRI	At Sea	-	-
SAT	Miami, Florida	8:00am	-

REVENUE BREAKDOWN

Category	Revenue	Contribution
Ticket Sales	$219,069,884.17	72.85%

METRICS

Each site can be branded to match your corporate identity and may contain up to five pages of content.

Using Sites

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

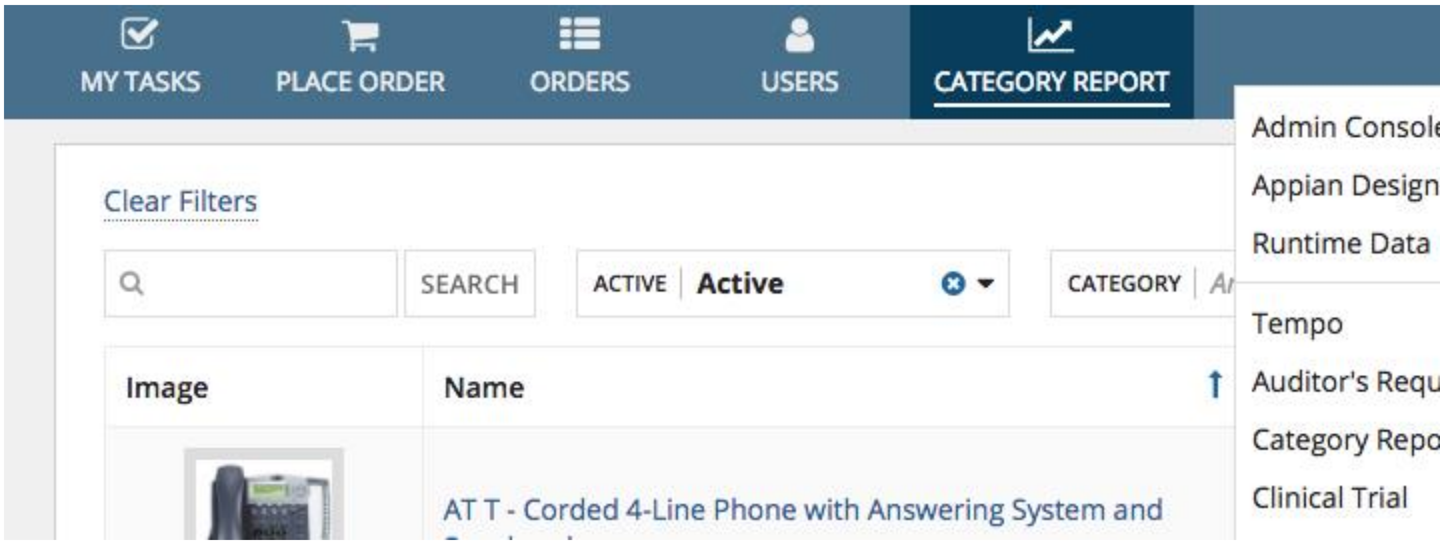
Users have a few ways to access sites:

- Directly by URL. Either the site or a specific site page can be targeted.
- Via the navigation menu in Tempo or another site. This will open the site in a new window and display the first page.

A site can be setup as the start page for users logging into Appian. To learn more about how to do this, see User Start Pages.

Navigation Menu

Using the site navigation menu, users can easily access other sites, Tempo, and other workspaces they have access to. The designer can also configure the menu to appear as the site name or as an icon. If the designer has configured the Tempo link to appear on the site navigation menu, it will always display at the top of the available sites. If the user only has access to their current site, the site navigation menu will not be shown.



User Menu

The site user menu provides users easy access to their profile, their user settings, or to sign out. Selecting profile will take the user to their user record. Selecting the user settings will open a dialog where they can update their user settings.

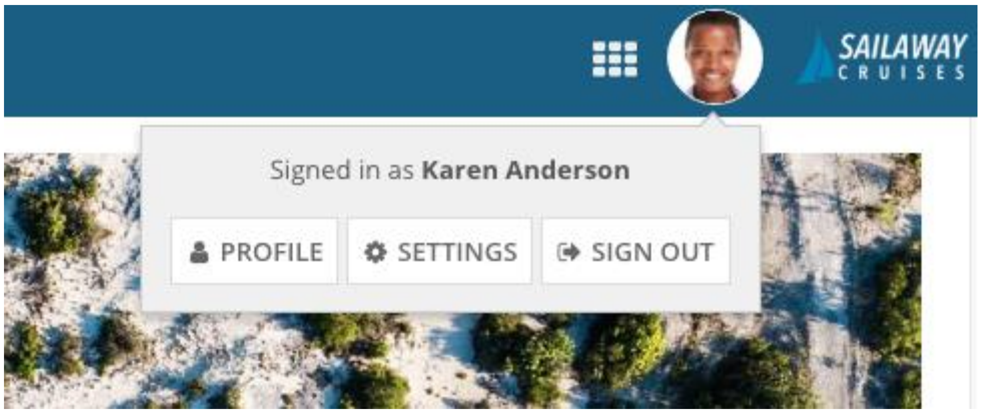
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Page Types

Pages are used to organize the content in a site. Users can navigate between site pages by clicking on the site page name or navigating via keyboard. To refresh the current site page a user is on, they can click or select the site page again. A site page can be configured to be one of three different types: action, record type, or report.

Actions

This page type allow users to initiate a process like submitting a new service request.

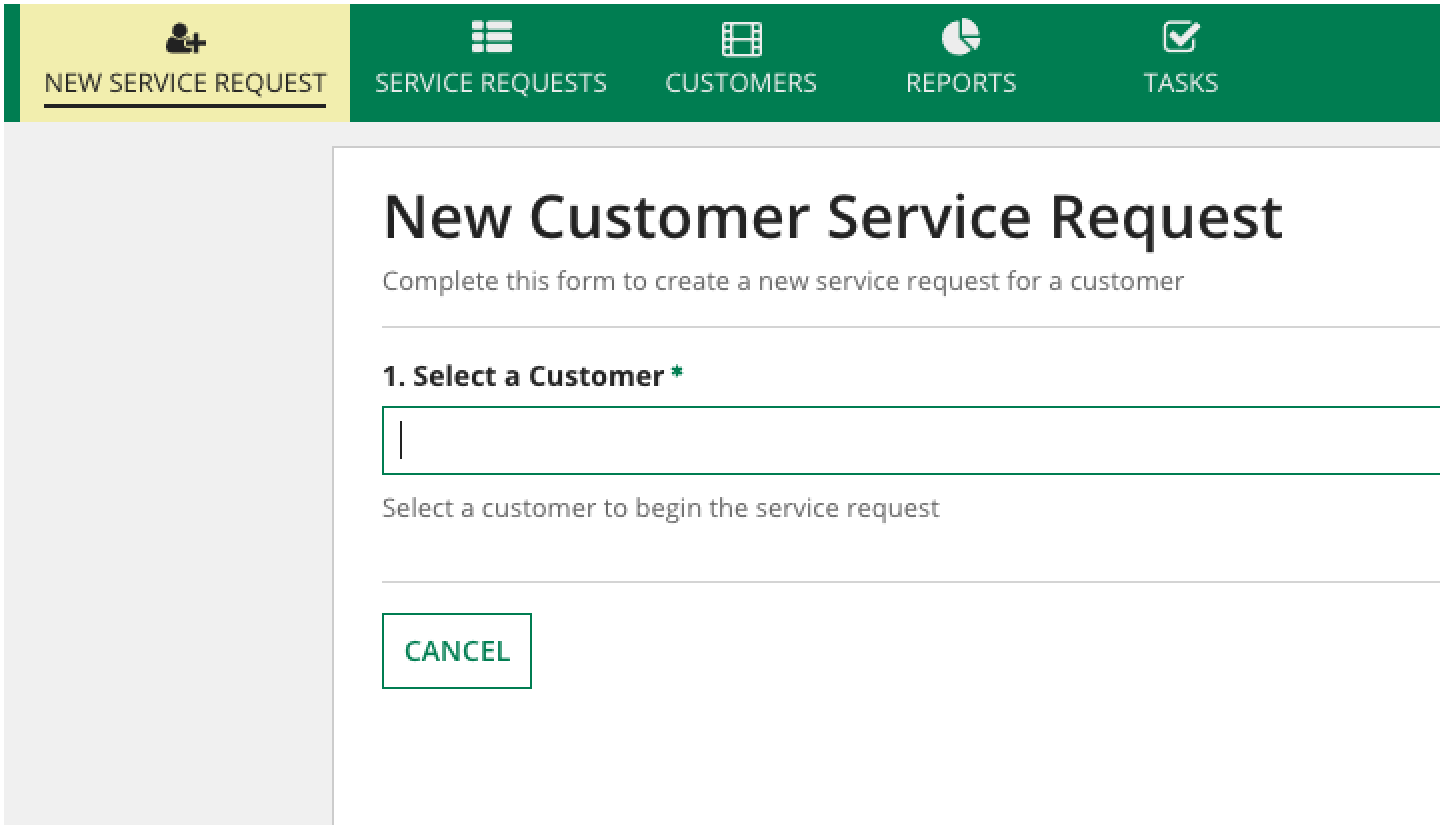
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



New Customer Service Request

Complete this form to create a new service request for a customer

1. Select a Customer *

Select a customer to begin the service request

Record Types

This page type provides users a list of records they can access for the selected record type. For example, showing a list of customers for the customer record type.

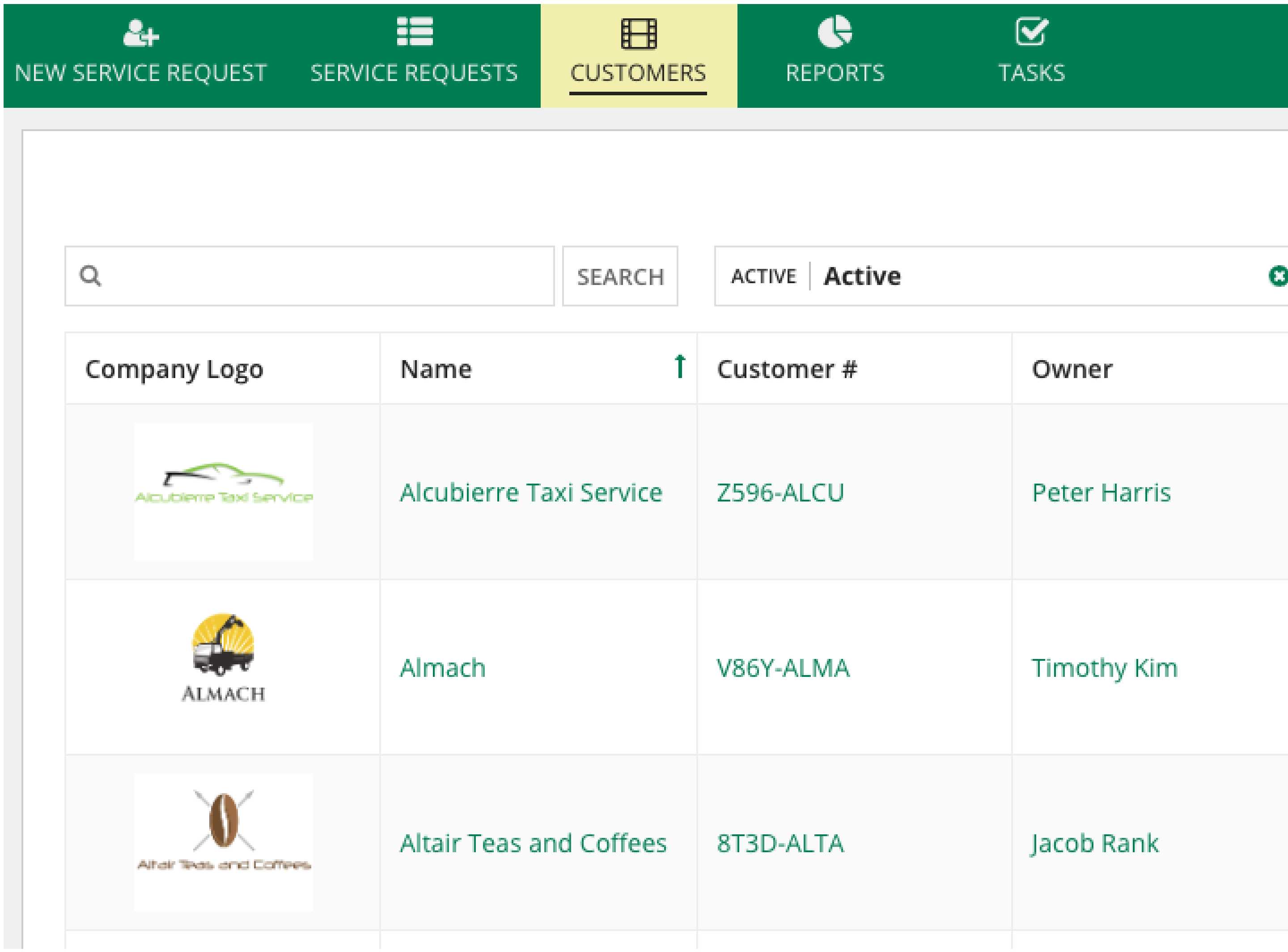
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a [record type](#) or [data store entity](#) using an intuitive interface. This page walks through the main components of the report builder.



Company Logo	Name	Customer #	Owner
	Alcubierre Taxi Service	Z596-ALCU	Peter Harris
	Almach	V86Y-ALMA	Timothy Kim
	Altair Teas and Coffees	8T3D-ALTA	Jacob Rank

Reports

This page type displays a specific report to the user as defined by the designer.

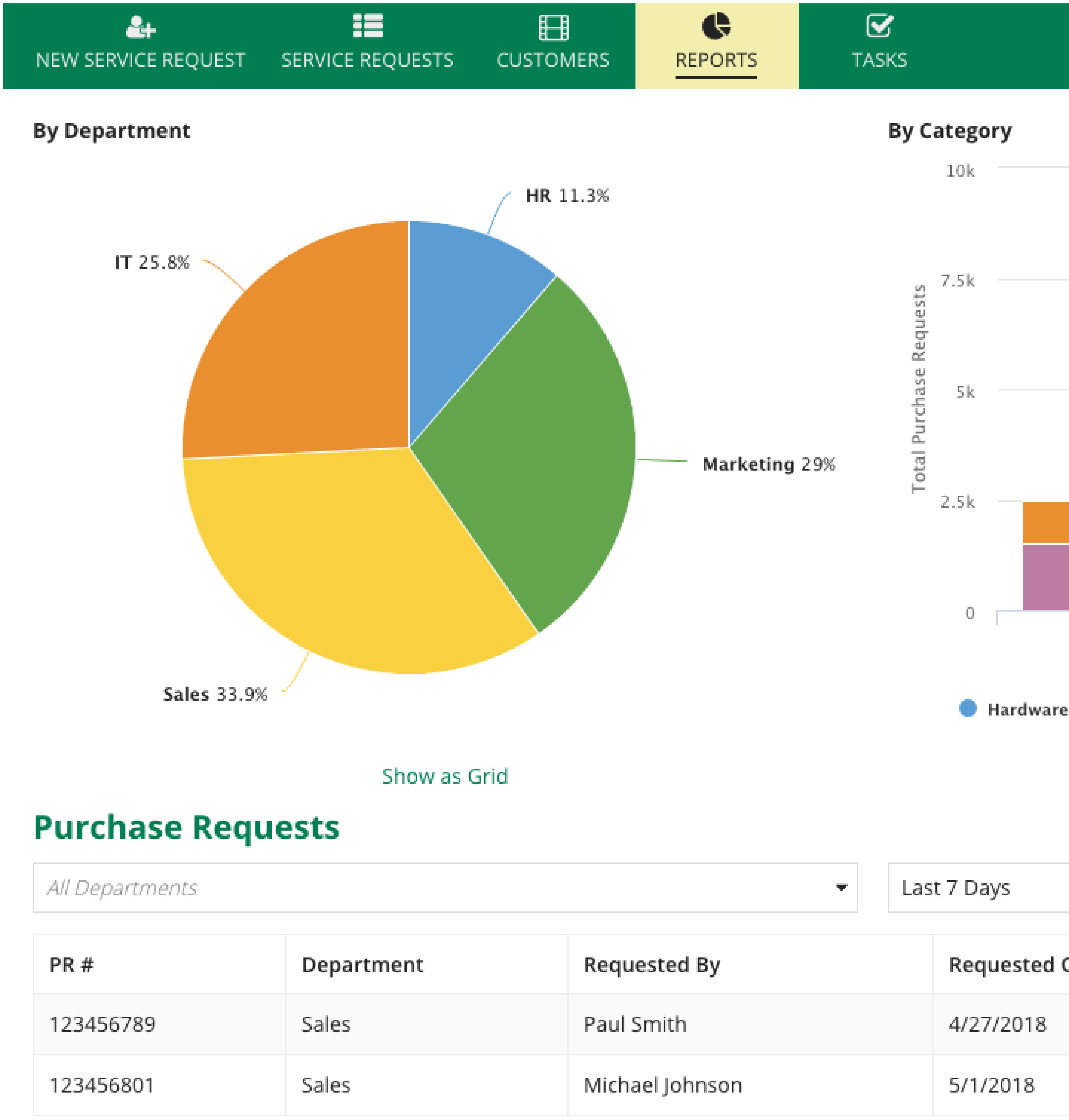
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Purchase Requests

PR #	Department	Requested By	Requested C
123456789	Sales	Paul Smith	4/27/2018
123456801	Sales	Michael Johnson	5/1/2018

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Task Notifications

Users that have a site as their defined start page will be presented with a task-specific version of their start page site when clicking on the task link in system generated task notifications. Once they complete the task, users will be directed to their start page.

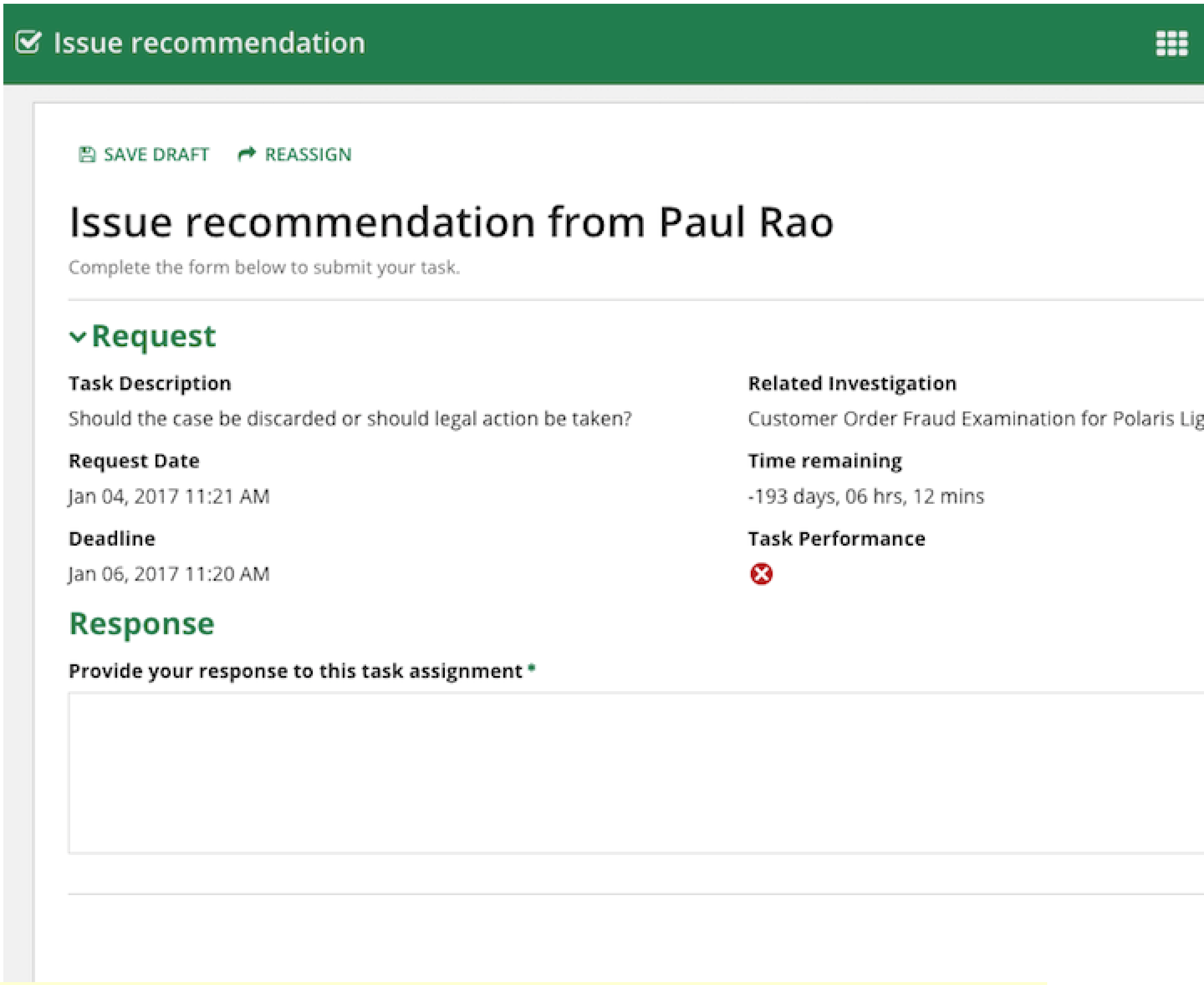
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Issue recommendation

SAVE DRAFT REASSIGN

Issue recommendation from Paul Rao

Complete the form below to submit your task.

Request

Task Description

Should the case be discarded or should legal action be taken?

Related Investigation

Customer Order Fraud Examination for Polaris Lig

Request Date

Jan 04, 2017 11:21 AM

Time remaining

-193 days, 06 hrs, 12 mins

Deadline

Jan 06, 2017 11:20 AM

Task Performance

Response

Provide your response to this task assignment *

Task notifications can be disabled at the environment level by administrators, at the task level by designers, or at the user level using notification settings.

Mobile

Use the Report Builder

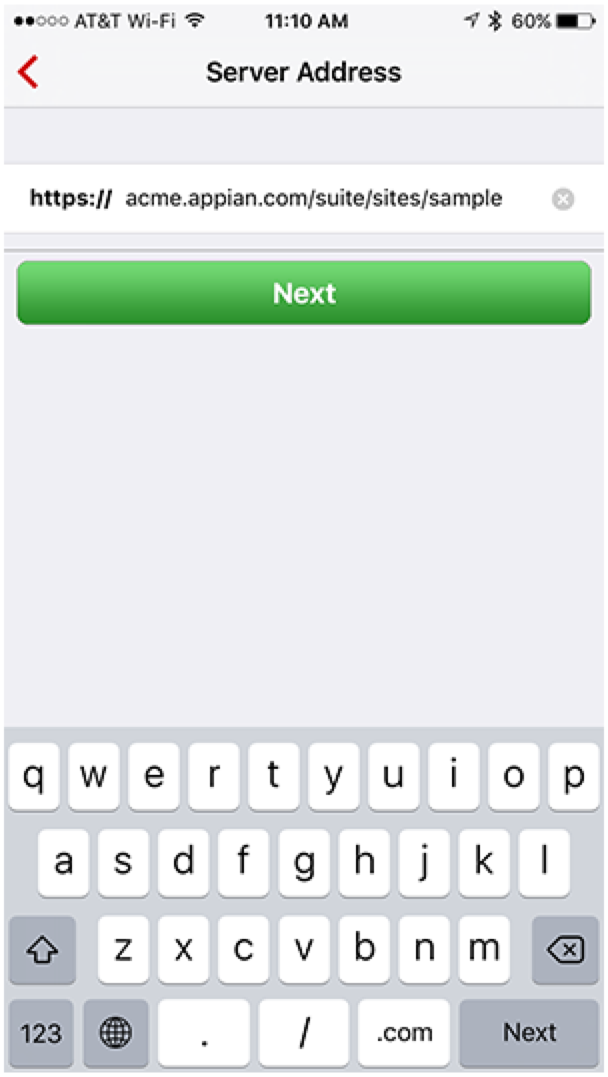
Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

To access your site on a mobile device, add an account to the Appian for Mobile Devices application with the full site URL as the server address.



●●○○○ AT&T Wi-Fi 11:10 AM 60%
Server Address
https:// acme.appian.com/suite/sites/sample
Next

Once you have added your account successfully, you will see the site.

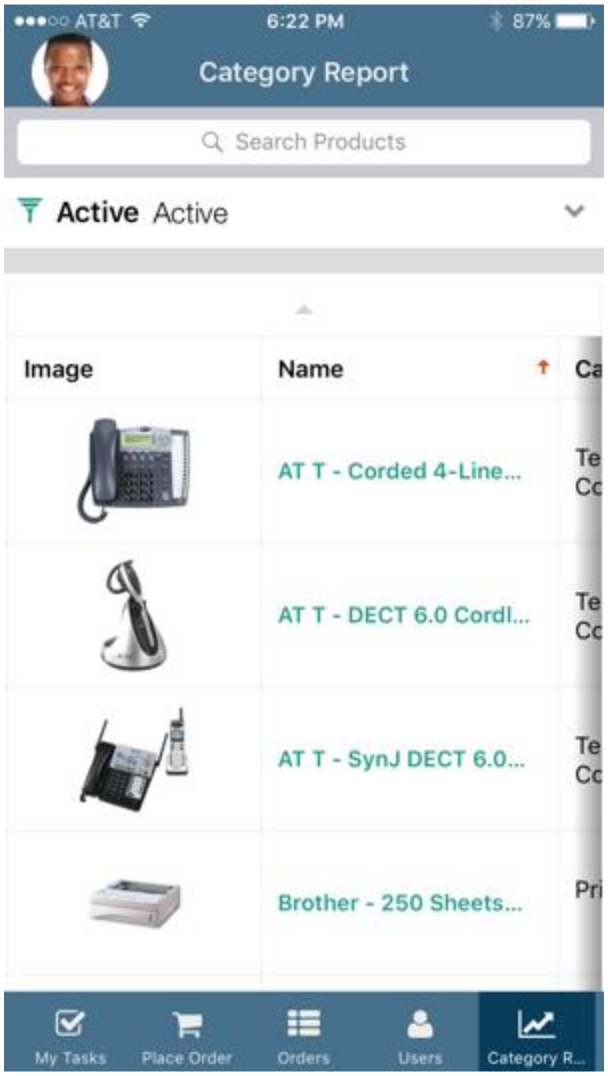
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Records Tutorial

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Overview

The walk-throughs on this page will help you create your first records. They are split up by source type and increase in complexity as you progress.

Use the data provided to understand how the configurations work. Then, try it with your own data. Keep in mind, the final configurations will need to change if your data has different field names.

The content below assumes a basic familiarity with interfaces and focuses more on the specifics of selecting a data source and modifying the list view and related actions for a record. Consider going through the Record Design page before proceeding.

See also: Expression-Backed Record Tutorial

Create the Appian Tutorial Application

The *Appian Tutorial* application is used to contain the design objects created while working through this tutorial.

The tutorial application only needs to be created once. If you have already created the tutorial application, skip the steps below.

To create the Appian Tutorial application

1. Log in to Appian Designer (for example, *myappiansite.com/suite/design*).
2. Click **New Application**.
3. In the **Name** field, type **Appian Tutorial**.
4. Optionally, in the **Description** field, add a short description.
5. Click **Create**.

The application contents view displays. Right now the application is empty. Each design object that you create during the course of this tutorial will appear in this list and be associated with the tutorial application.

Create Entity-Backed Records

The steps below show how to create a simple entity-backed record type for employee information saved to a data store entity.

Before we begin, let's create the data store entity and its data:

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

1. Create a custom data type called **Employee** with the following fields in the associated data types:
    - id (Integer)
    - firstName (Text)
    - lastName (Text)
    - department (Text)
    - title (Text)
    - phoneNumber (Text)
    - startDate (Date)

      See also: Custom Data Type
2. Designate the id field as the primary key and set to generate value. Only entities that have a defined primary key (and not an automatically generated one) can be a record source.
    - See also: Primary Keys
3. Save and publish the CDT.
4. Create a data store entity `Employees` that references the `Employee` CDT and publish its data store.
    - See also: Data Stores
5. Insert the following values into the table:

id	firstName	lastName	department	title	phoneNumber	startDate
1	John	Smith	Engineering	Director	555-123-4567	2013-01-02
2	Michael	Johnson	Finance	Analyst	555-987-6543	2012-06-15
3	Mary	Reed	Engineering	Software Engineer	555-456-0123	2001-01-02
4	Angela	Cooper	Sales	Manager	555-123-4567	2005-10-15
5	Elizabeth	Ward	Sales	Sales Associate	555-987-6543	2010-01-02

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

id	firstName	lastName	department	title	phoneNumber	startDate
6	Daniel	Lewis	HR	Manager	555-876-5432	2010-01-02
7	Paul	Martin	Finance	Analyst	555-609-3691	2009-12-01
8	Jessica	Peterson	Finance	Analyst	555-987-6543	2004-11-01
9	Mark	Hall	Professional Services	Director	555-012-3456	2009-06-01
10	Rebecca	Wood	Engineering	Manager	555-210-3456	2008-07-27
11	Pamela	Sanders	Engineering	Software Engineer	555-123-4567	2005-04-01
12	Christopher	Morris	Professional Services	Consultant	555-456-7890	2011-03-29
13	Kevin	Stewart	Professional Services	Manager	555-345-6789	2008-02-29
14	Stephen	Edwards	Sales	Sales Associate	555-765-4321	2006-01-02

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

id	firstName	lastName	department	title	phoneNumber	startDate
15	Janet	Coleman	Finance	Director	555-654-3210	1999-12-30
16	Scott	Bailey	Engineering	Software Engineer	555-678-1235	2005-03-15
17	Andrew	Nelson	Professional Services	Consultant	555-789-4560	2005-03-15
18	Michelle	Foster	HR	Director	555-345-6789	2005-03-15
19	Laura	Bryant	Sales	Sales Associate	555-987-6543	2004-11-01
20	William	Ross	Engineering	Software Engineer	555-123-4567	2008-07-27

Now that we have the data, we can create the record type which will generate a record for each of the rows in our Employees data store entity.

1. Navigate to the application contents view of the **Appian Tutorial** application (if needed).
2. Click **New**, and then click **Record Type**.
3. In the **Create Record Type** dialog, enter the following information:
   - **Name**: `Employee Directory`
   - **Plural Name**: `Employees`
   - **Description**: `List of current employees`
4. Click **Create & Edit**.

After the Employee Directory record type opens in the record type designer, we need to add some more information:

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

1. Enter the following values into the Data section:
   - **Source**: `Data Store Entity`
   - **Data Store**: `Employees`
   - **Entity**: `Employee`
2. In the Record View Details section, enter the following for the **Record Title** field:

```
rf!firstName & " " & rf!lastName
```

3. In the **Views** grid, click **Summary**, then paste the following expression into the **Interface** field:

```
={
    a!textField(
      label: "Name",
      labelPosition: "ADJACENT",
      value: rf!firstName & " " & rf!lastName,
      readOnly: true
    ),
    a!textField(
      label: "Department",
      labelPosition: "ADJACENT",
      value: rf!department,
      readOnly: true
    ),
    a!textField(
      label: "Title",
      labelPosition: "ADJACENT",
      value: rf!title,
      readOnly: true
    ),
    a!textField(
      label: "Contact Number",
      labelPosition: "ADJACENT",
      value: rf!phoneNumber,
      readOnly: true
```

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

```
    ),
    a!textField(
      label: "Start Date",
      labelPosition: "ADJACENT",
      value: rf!startDate,
      readOnly: true
    )
}
```

4. Click **Save** in the header of the record type designer.

You have just created a record for each of the employees listed in the Employees data store entity.

To view your records, open Tempo in a new browser, select the Records tab, and click Employees. You should see the following:

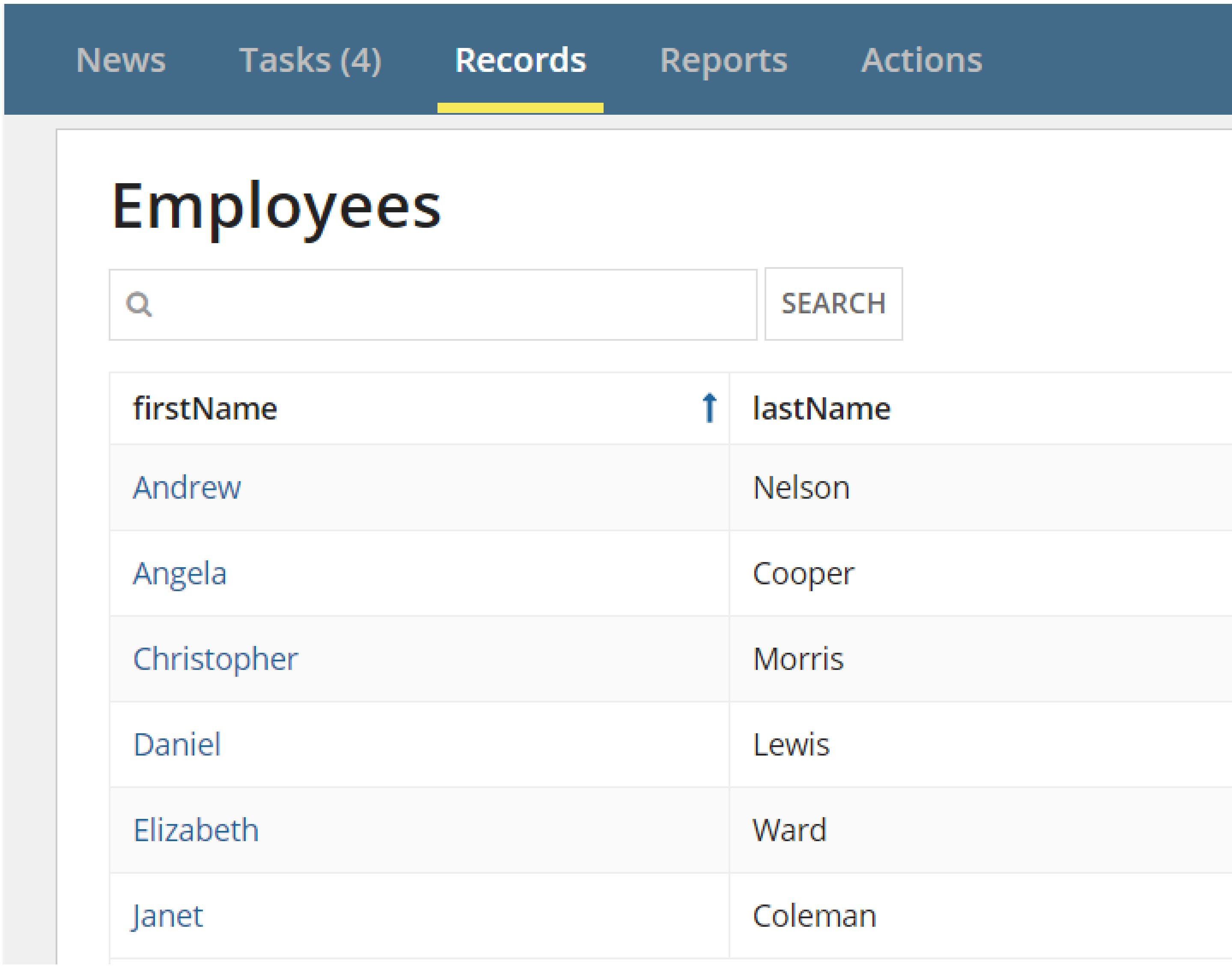
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Employees

firstName	lastName
Andrew	Nelson
Angela	Cooper
Christopher	Morris
Daniel	Lewis
Elizabeth	Ward
Janet	Coleman

To view a record, select a name from the list. For Andrew Nelson you should see the following:

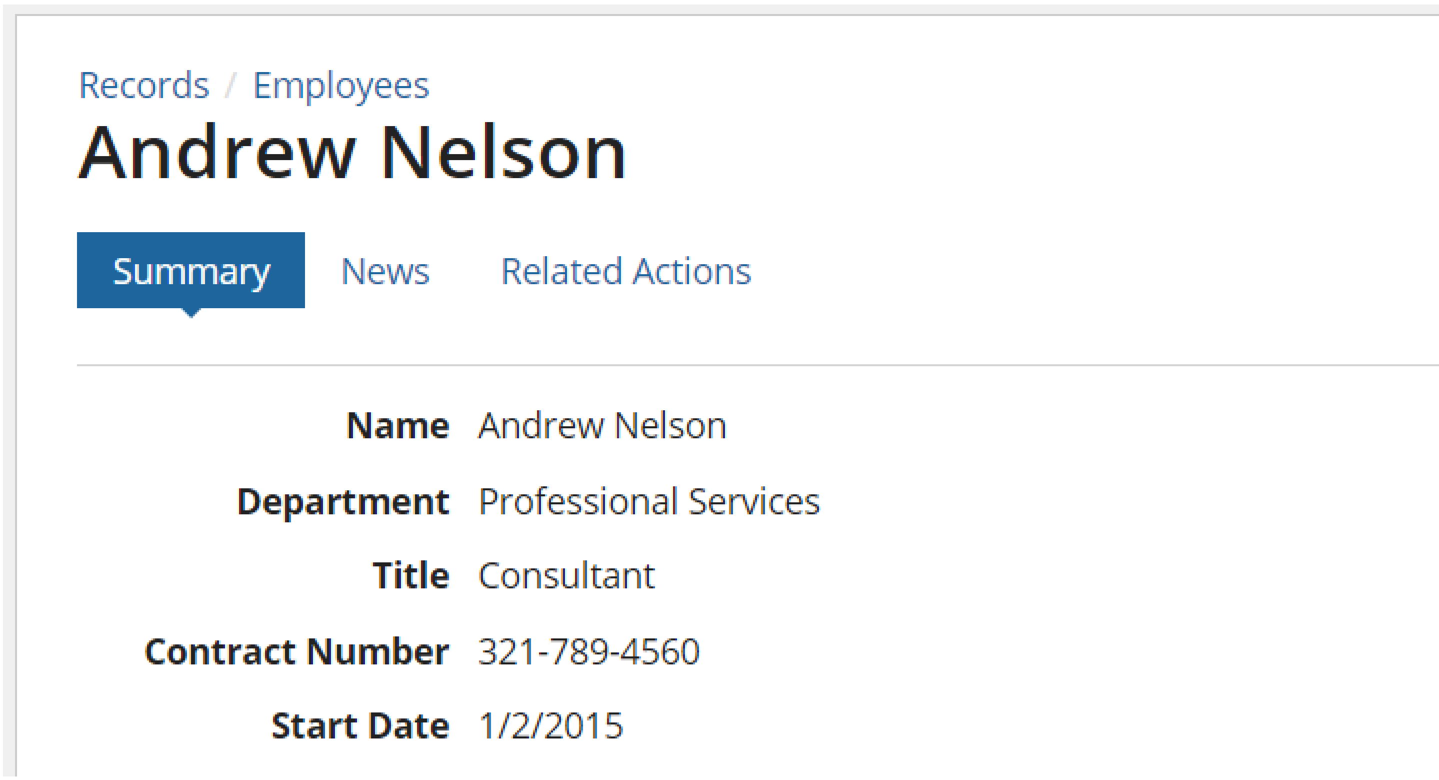
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Records / Employees

Andrew Nelson

Name	Andrew Nelson
Department	Professional Services
Title	Consultant
Contract Number	321-789-4560
Start Date	1/2/2015

The design of the Summary view is based on the expression we added to the Interface field for the Summary view of the record type. Looking back at the expression, you'll notice we used the `rf!` domain to access our data. This is required for entity-backed records. Other than that, you can change this record view design as you would any other interface.

You can also configure additional record views for users to view more information on the record.

See also: Create a Record View

Since this record type is not backed by a process model with quick tasks and we have not configured a related action yet, selecting the Related Actions view displays an empty list. Later on, we'll add to this list, but for now, let's focus on providing users with more data.

While working through the sections below, keep Tempo open with your record in one browser and the record type designer interface with your record type open in another. This way, when we make changes to the record type, you can just refresh Tempo to see the changes automatically.

Modify the Record List

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Currently, our record list is using the defaults. While this provides us with a lot of information, it's not exactly what we want to see.

Let's start by removing the Id column.

1. With the Employee Directory record type still open, scroll down to the Record List section.
2. Click **Edit Record List**. You should see the following:

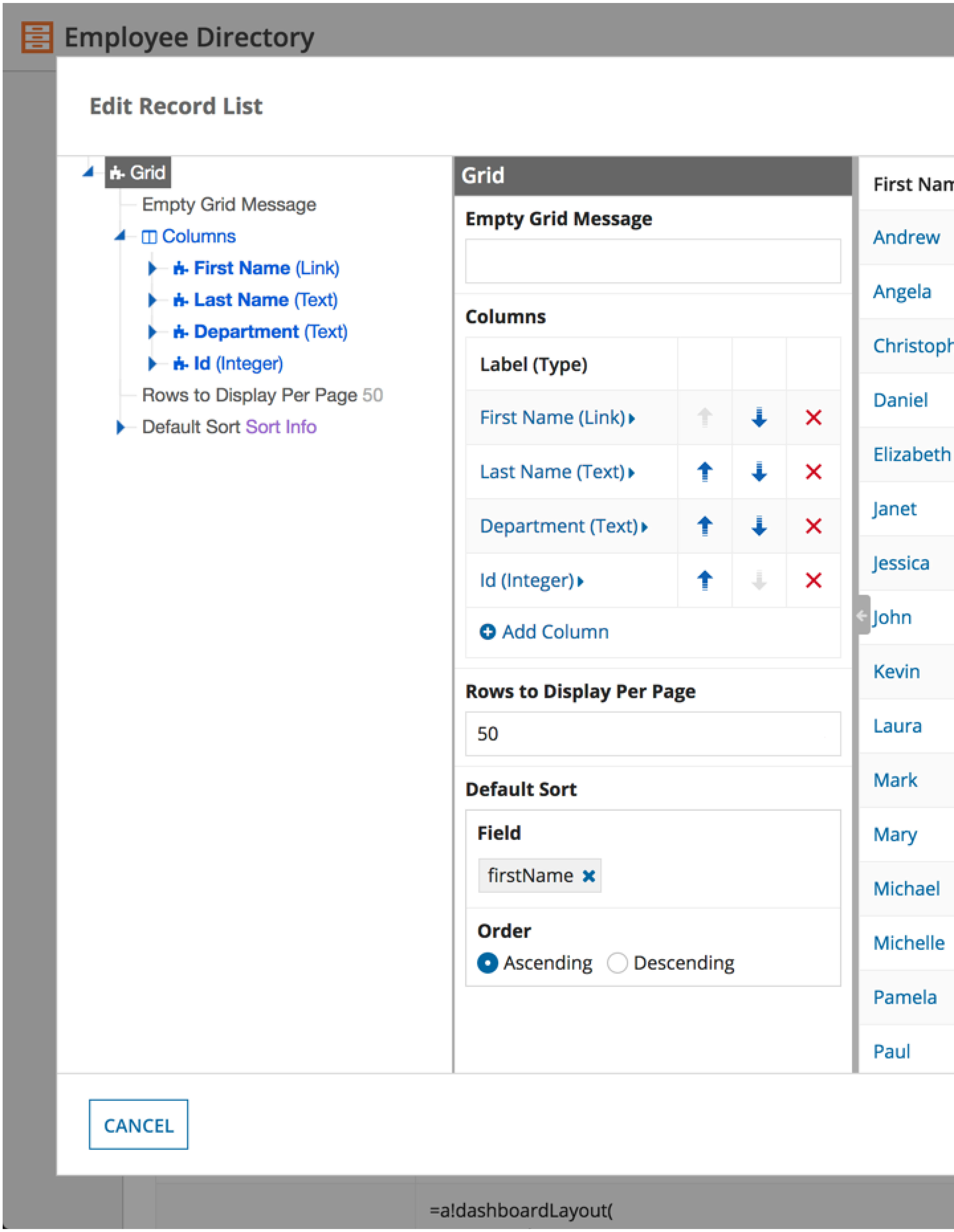
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Employee Directory

Edit Record List

- Grid
  - Empty Grid Message
  - Columns
    - **First Name** (Link)
    - **Last Name** (Text)
    - **Department** (Text)
    - **Id** (Integer)
  - Rows to Display Per Page 50
  - Default Sort Sort Info

Grid

Empty Grid Message

Columns

Label (Type)			
First Name (Link) ▸	↑	↓	✕
Last Name (Text) ▸	↑	↓	✕
Department (Text) ▸	↑	↓	✕
Id (Integer) ▸	↑	↓	✕

Add Column

Rows to Display Per Page

50

Default Sort

Field

firstName ✕

Order

[✓ Ascending] Descending

First Nam
Andrew
Angela
Christoph
Daniel
Elizabeth
Janet
Jessica
John
Kevin
Laura
Mark
Mary
Michael
Michelle
Pamela
Paul

CANCEL

=a!dashboardLayout(

1. Remove the Id column.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

You should see the
following:

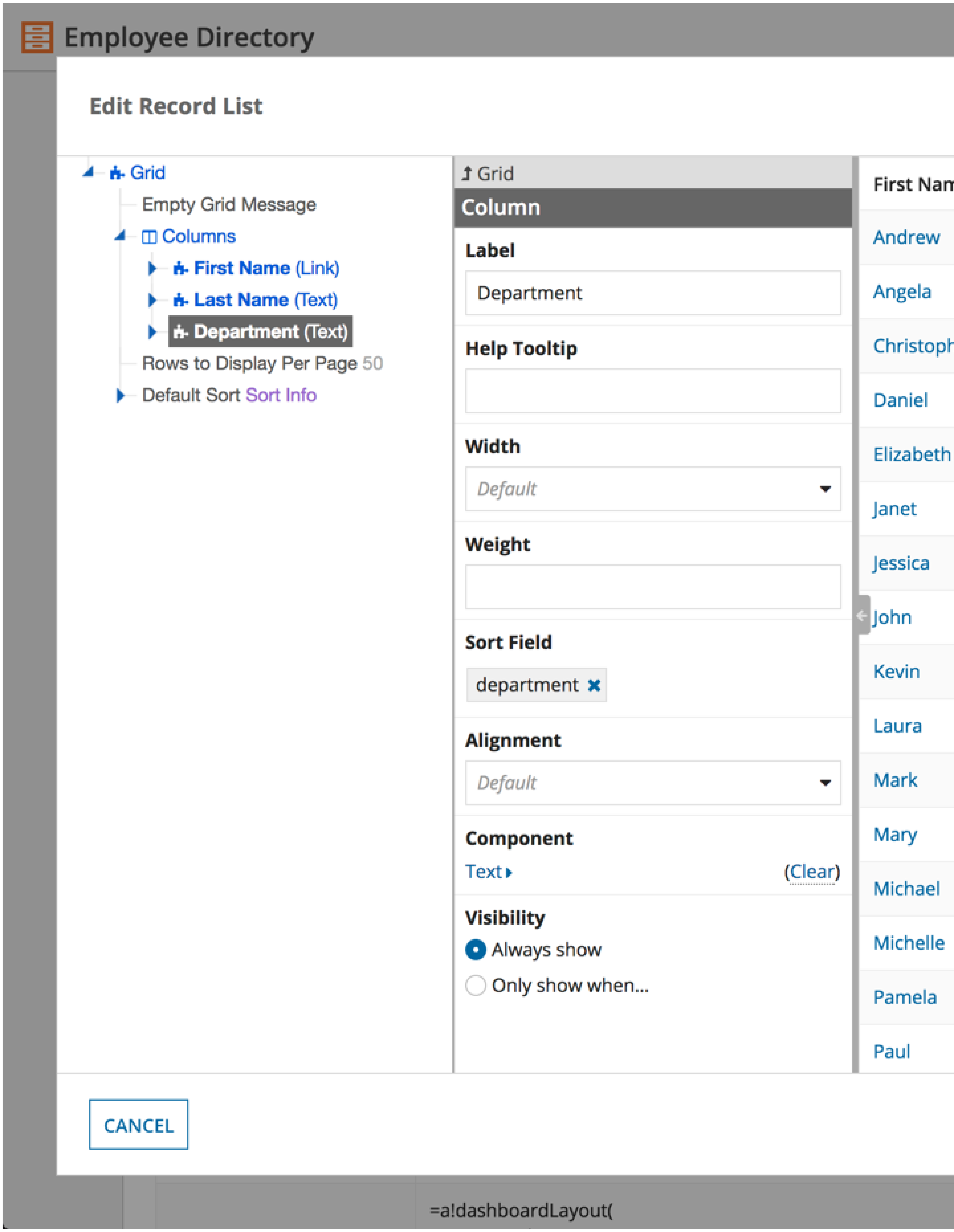
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Employee Directory

Edit Record List

- Grid
  - Empty Grid Message
  - Columns
    - First Name (Link)
    - Last Name (Text)
    - Department (Text)
  - Rows to Display Per Page 50
  - Default Sort Sort Info

Grid

Column

Label

Department

Help Tooltip

Width

Default

Weight

Sort Field

department

Alignment

Default

Component

Text (Clear)

Visibility

● Always show

○ Only show when...

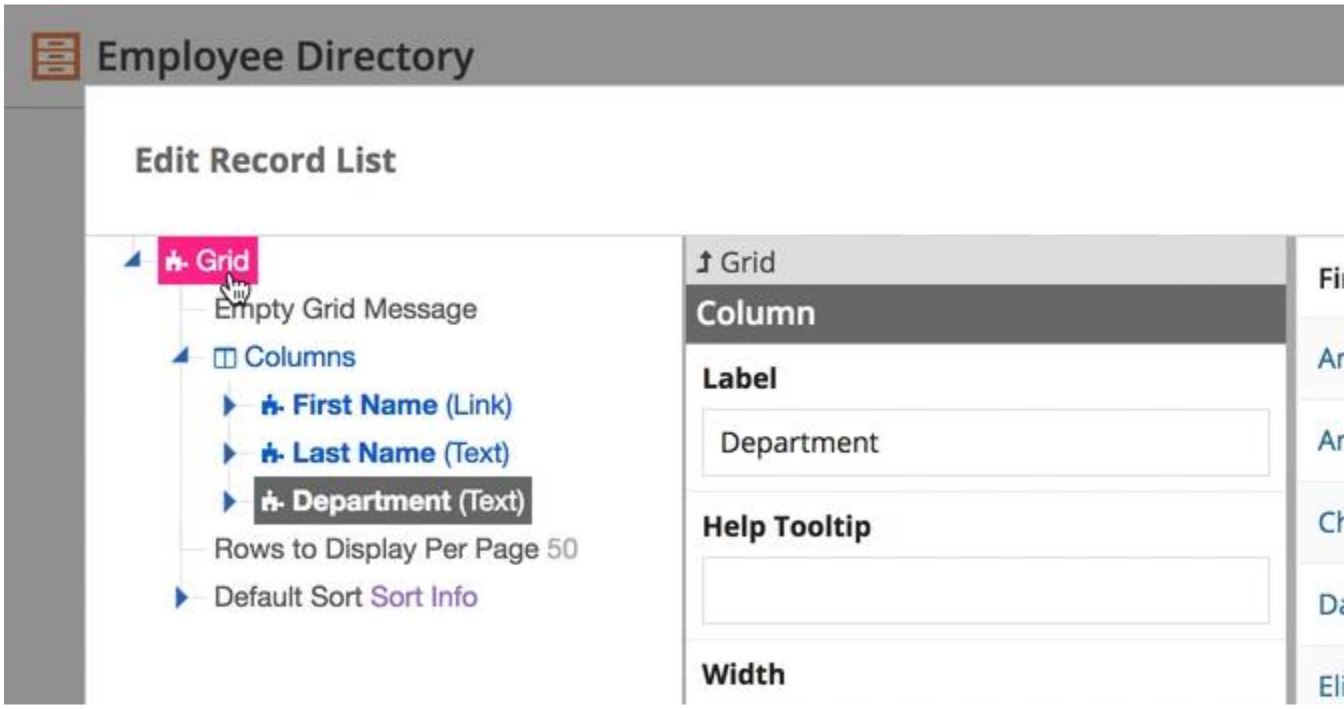
First Nam
Andrew
Angela
Christoph
Daniel
Elizabeth
Janet
Jessica
John
Kevin
Laura
Mark
Mary
Michael
Michelle
Pamela
Paul

CANCEL

```
=a!dashboardLayout(
```

Now let's add a new column to display the employee's title.

1. Go back to the grid by clicking on **Grid**, as shown below:



2. Add a new column using the **+ Add Column** link.
3. Configure the following properties for the new column:
   - **Label**: `Title`
   - **Sort Field**: `title`
4. Select the component for the column by clicking the **Select Component** link and clicking on **Text** in the dialog.
5. Configure the following property for the text component:
   - Display Value: rf!title

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

It should look like the following:

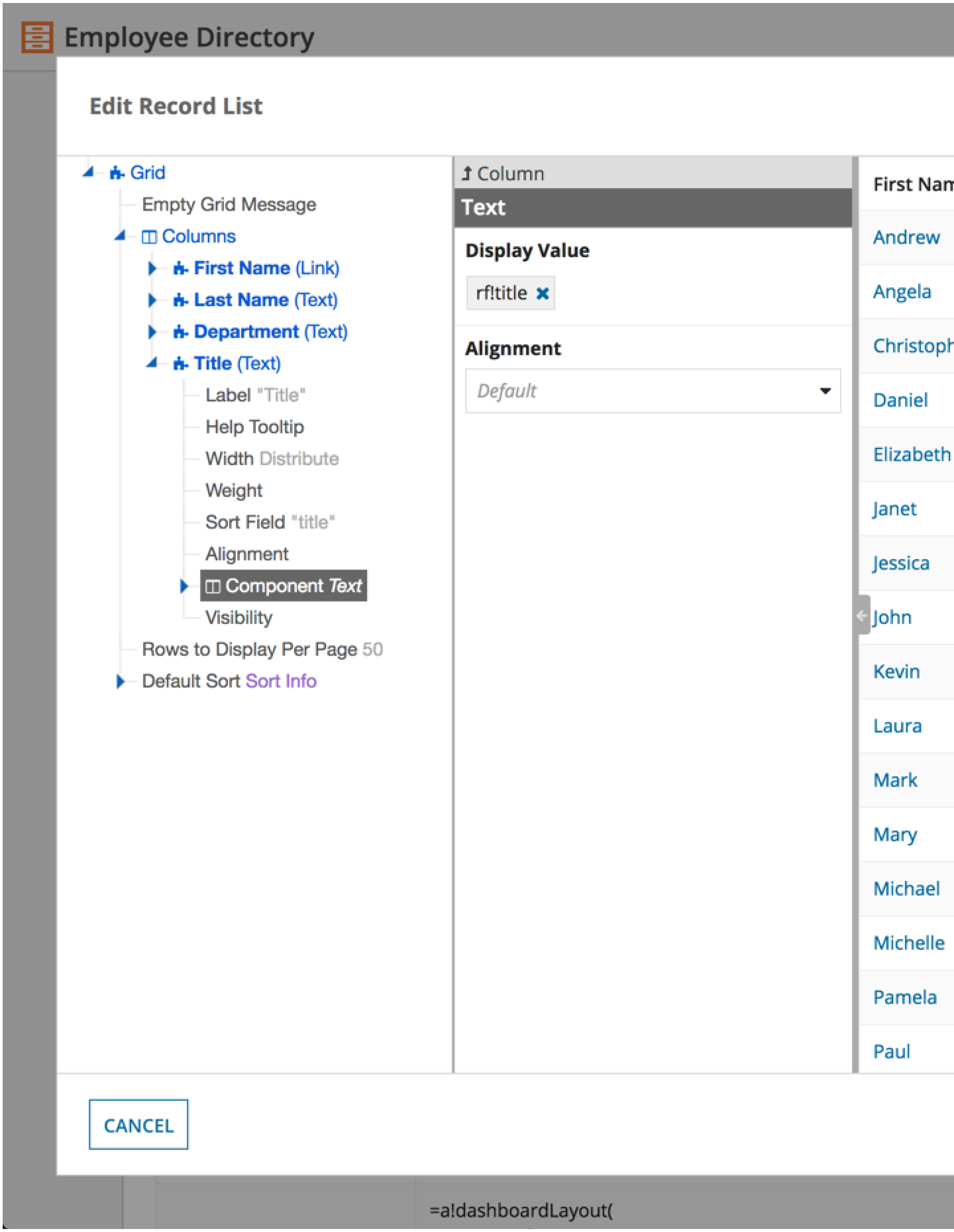
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Employee Directory

Edit Record List

- Grid
  - Empty Grid Message
  - Columns
    - First Name (Link)
    - Last Name (Text)
    - Department (Text)
    - Title (Text)
      - Label "Title"
      - Help Tooltip
      - Width Distribute
      - Weight
      - Sort Field "title"
      - Alignment
      - Component *Text*
      - Visibility
  - Rows to Display Per Page 50
  - Default Sort Sort Info

Column

Text

Display Value

rf!title

Alignment

Default

First Nam
Andrew
Angela
Christoph
Daniel
Elizabeth
Janet
Jessica
John
Kevin
Laura
Mark
Mary
Michael
Michelle
Pamela
Paul

CANCEL

=a!dashboardLayout(

Finally, let's combine the firstName and lastName columns into a single column.

1. Go back to the grid and remove the lastName column.
2. Update the following properties of the firstName column:
    - **Label**: `Name`
    - **Sort Field**: `lastName`

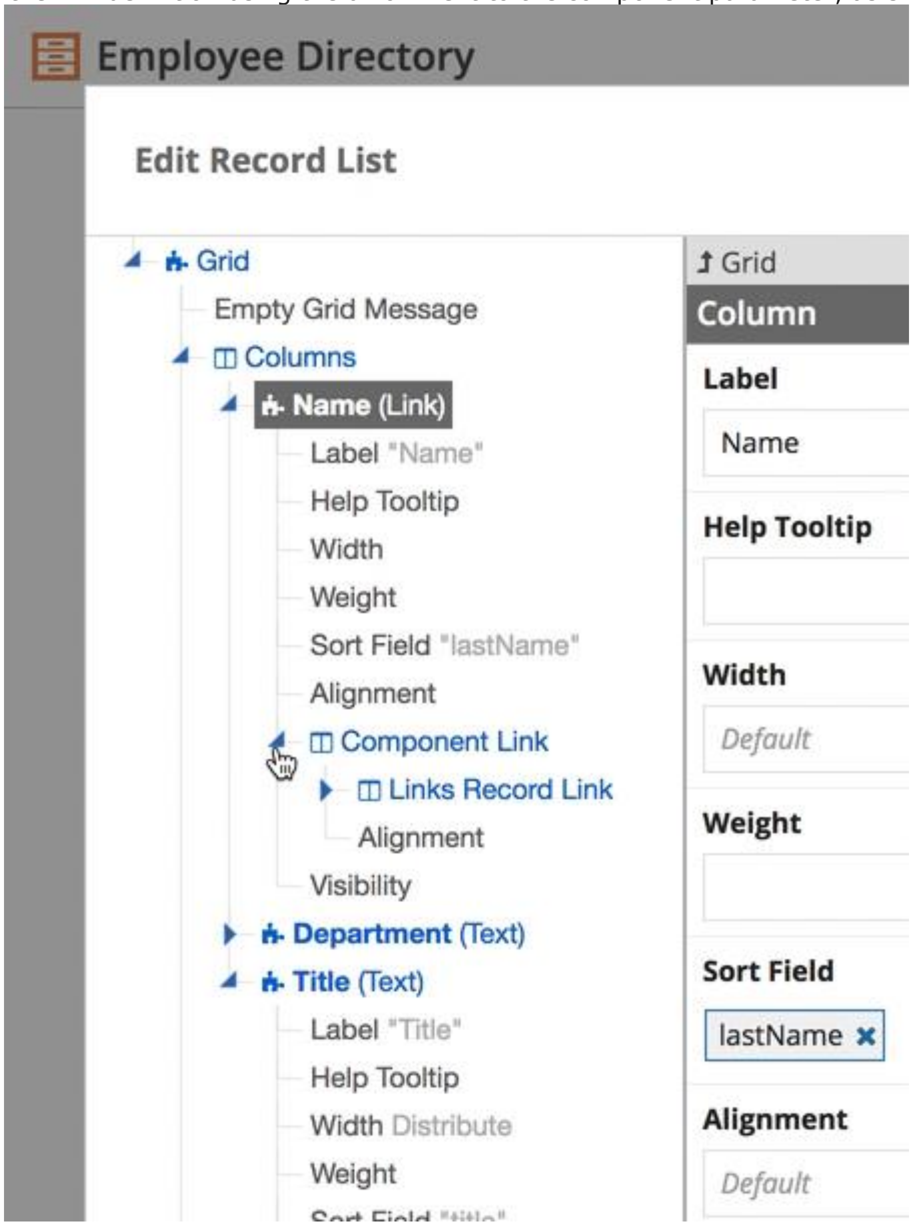
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

3. Expand the link definition using the arrow next to the component parameter, as shown below: 

4. Click on **Record Link** to view the link configuration and update the following property:
   - Label: `rf!firstName & " " & rf!lastName`

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

It should look like the following:

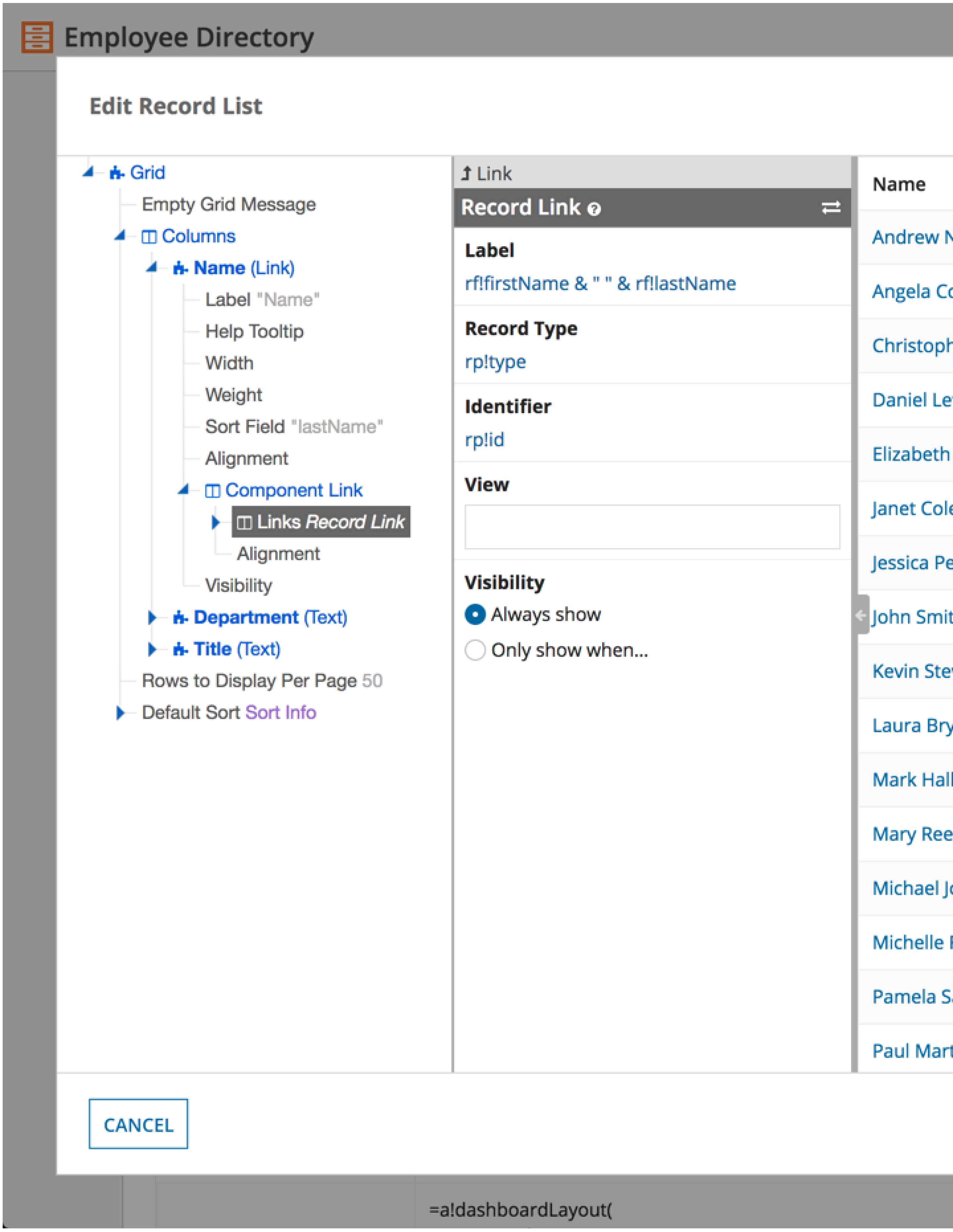
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Employee Directory

Edit Record List

Grid
- Empty Grid Message
- Columns
  - Name (Link)
    - Label "Name"
    - Help Tooltip
    - Width
    - Weight
    - Sort Field "lastName"
    - Alignment
    - Component Link
      - Links *Record Link*
      - Alignment
    - Visibility
  - Department (Text)
  - Title (Text)
- Rows to Display Per Page 50
- Default Sort Sort Info

Link

Record Link

Label
rf!firstName & " " & rf!lastName

Record Type
rp!type

Identifier
rp!id

View

Visibility
[✓ Always show]
Only show when...

Name
Andrew N
Angela Co
Christoph
Daniel Le
Elizabeth
Janet Cole
Jessica Pe
John Smit
Kevin Ste
Laura Bry
Mark Hall
Mary Ree
Michael J
Michelle
Pamela S
Paul Mart

CANCEL

=a!dashboardLayout(

4. Click **OK** at the bottom of the dialog and save the record.

For more information on the different options available for configuring a record list, see Create a Record List.

Add A User Filter

Now that we have Employee Directory records, let's add a user filter so users can filter the records by Department.

1. With the Employee Directory record type still open, scroll down to the User Filters section.
2. Click **+ New User Filter** below the User Filters grid.
3. Enter the following values for the associated fields exactly as shown (including quotes where present):
   - Filter Name: Department
   - Filter Field: department
   - Filter Label: `"Department"`
4. Click **+ New Filter Option** below the Filter Options grid.
5. Enter the following values for the associated fields exactly as shown (including quotes where present):
   - Option Label: `"Engineering"`
   - Operator: =
   - Value: `"Engineering"`
6. Click **Save Filter Option**.
   - This creates the first filter option of Engineering for the Department user filter. Now we need to add filter options for the remaining departments.
7. Following the same steps, create the remaining filter options:

Option Label	Operator	Value
`Finance`	=	`Finance`
`Human Resources`	=	`HR`

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Option Label	Operator	Value
`Professional Services`	=	`Professional Services`
`Sales`	=	`Sales`

8. Click **OK** in the Create New User Filter modal dialog.
9. Click **Save** in the header of the Record Type Designer.

You have just created a Department user filter with the five departments as options.

Refresh the Employees record in Tempo. You should see the following:

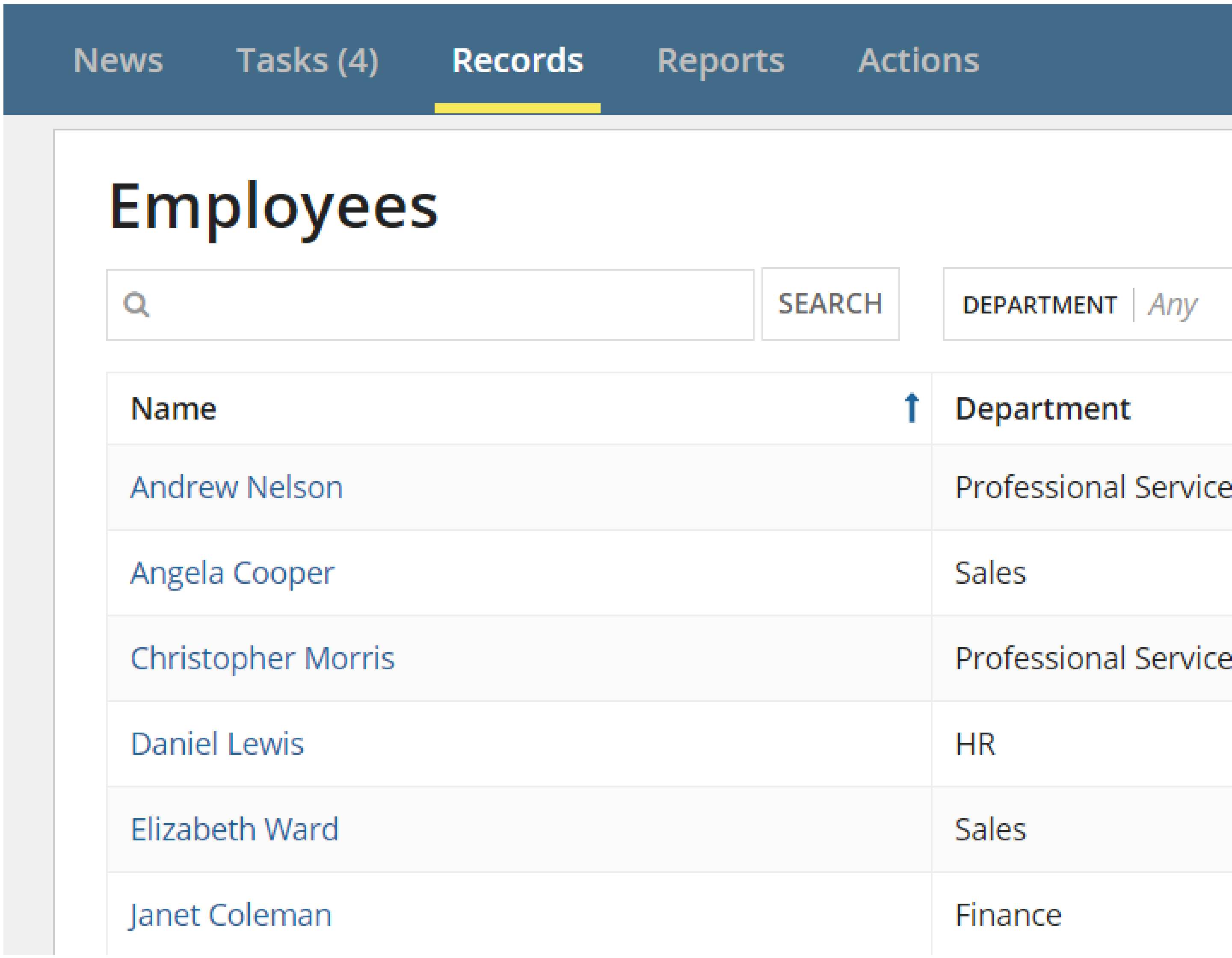
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.



Employees

Name	Department
Andrew Nelson	Professional Service
Angela Cooper	Sales
Christopher Morris	Professional Service
Daniel Lewis	HR
Elizabeth Ward	Sales
Janet Coleman	Finance

To see the full list of options, click the **Department** filter dropdown. Users can select any one of these options to filter the records down to just those in the selected department(s).

For more information on how to configure user filters, see Create User Filters.

Add a Related Action

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

Remember how the Related Actions view for each record was empty? Now let's add an action to it that allows users to update the phone number for the record they are currently viewing.

1. Create a query rule called `getEmployeeById` that takes an integer as a parameter and returns the Employee with that id.
2. Create a process model called Update Phone Number with a process variable named `employeeId` of type Number (Integer) marked as a parameter and no start form.
3. Set up an activity chain from the model's start node into a User Input Activity.
4. For the User Input activity, assign the task to "=pp!initiator", and create a form titled "Update Phone Number" with one text field called "New Phone Number" that has a default value of `rule!getEmployeeById(pv!employeeId).phoneNumber` and saves into `ac!newPhoneNumber`. `ac!newPhoneNumber`should then be saved into a process variable so that it can be accessed by other nodes later in the process.
5. Use the Write to Data Store Entity Smart Service to update the data store entity that has an id that matches the `employeeId` process variable.
6. Publish the process model.
7. Navigate to the Employee Record Type and open it.
8. Scroll down to the Related Actions section, and click **+ New Related Action**.
9. Add a display name and description

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

10. For Process Model, select the process model you configured in step 2. You should see the following:

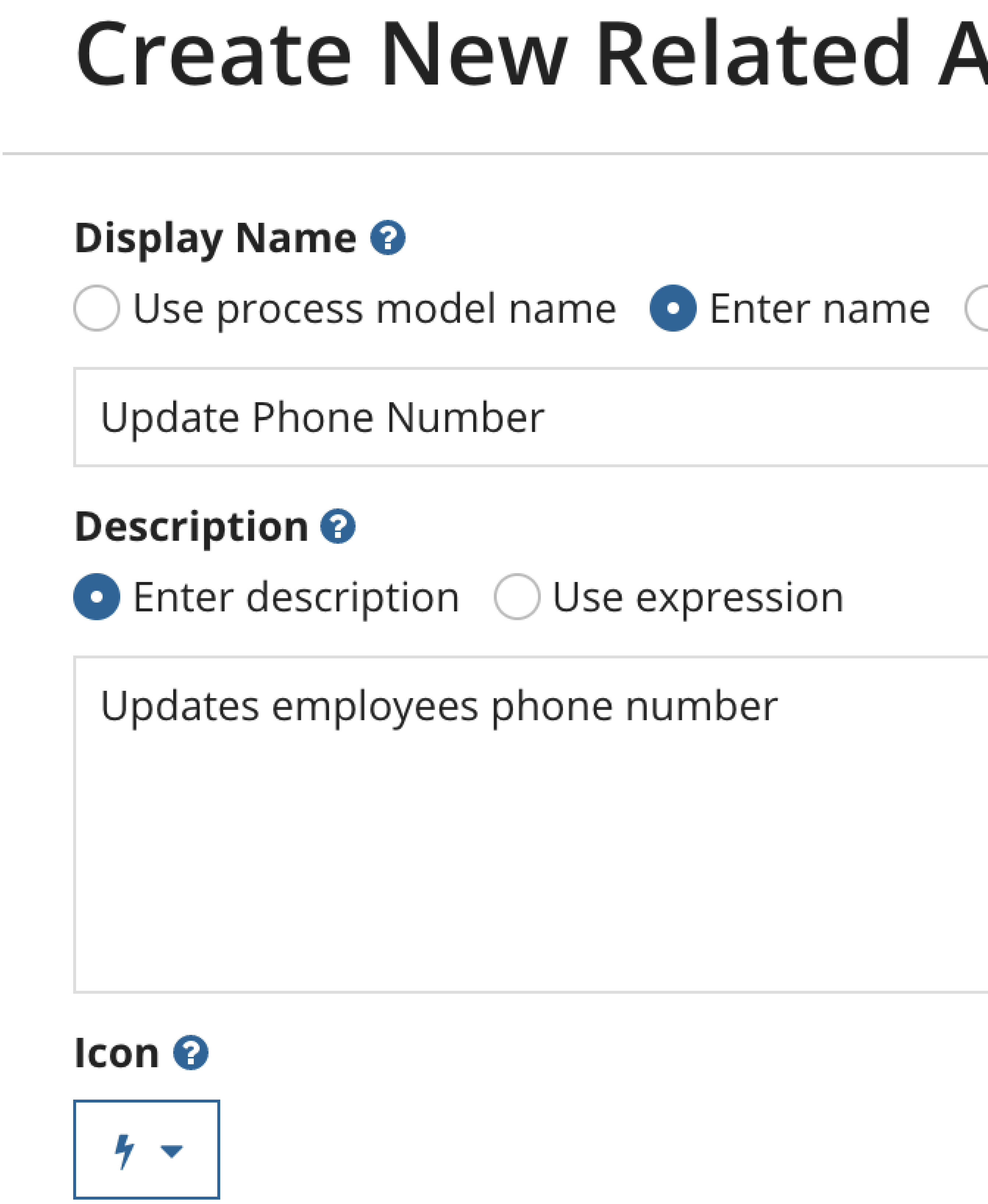
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Create New Related A

Display Name
Use process model name [✓ Enter name]

Update Phone Number

Description
[✓ Enter description] Use expression

Updates employees phone number

Icon

U

Ov

This

Afte
get

The
type
the

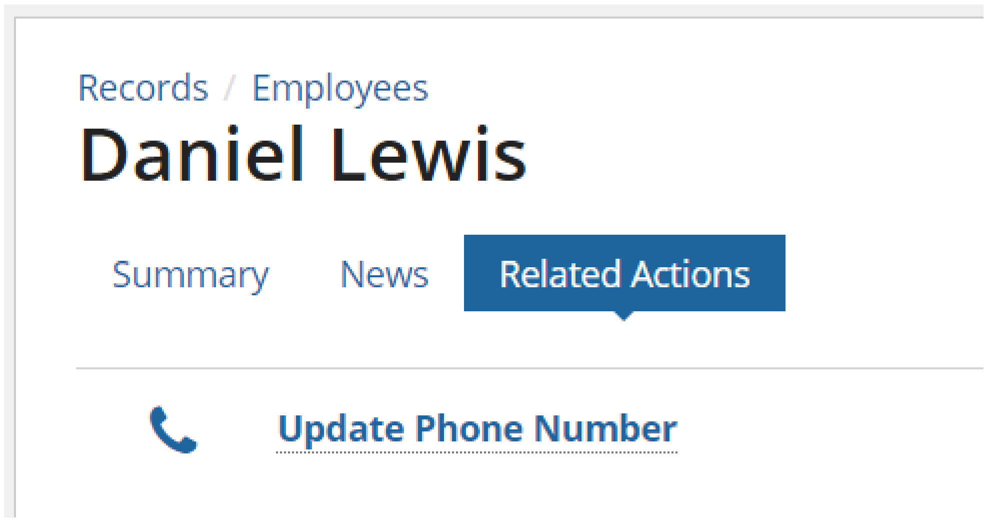
11. In the **Context** field, change the value to:

```
{
  employeeId: rp!id /* Number (Integer) */
}
```

12. Click **OK**.

The *Update Phone Number* process model is now a related action for your *Employee Directory* records where users can update the phone number of the record they're viewing.

Refresh the *Employees* record in Tempo and select an employee from the list. Click **Related Actions**. You should see the following:



Click **Update Phone Number**. You should see the following:

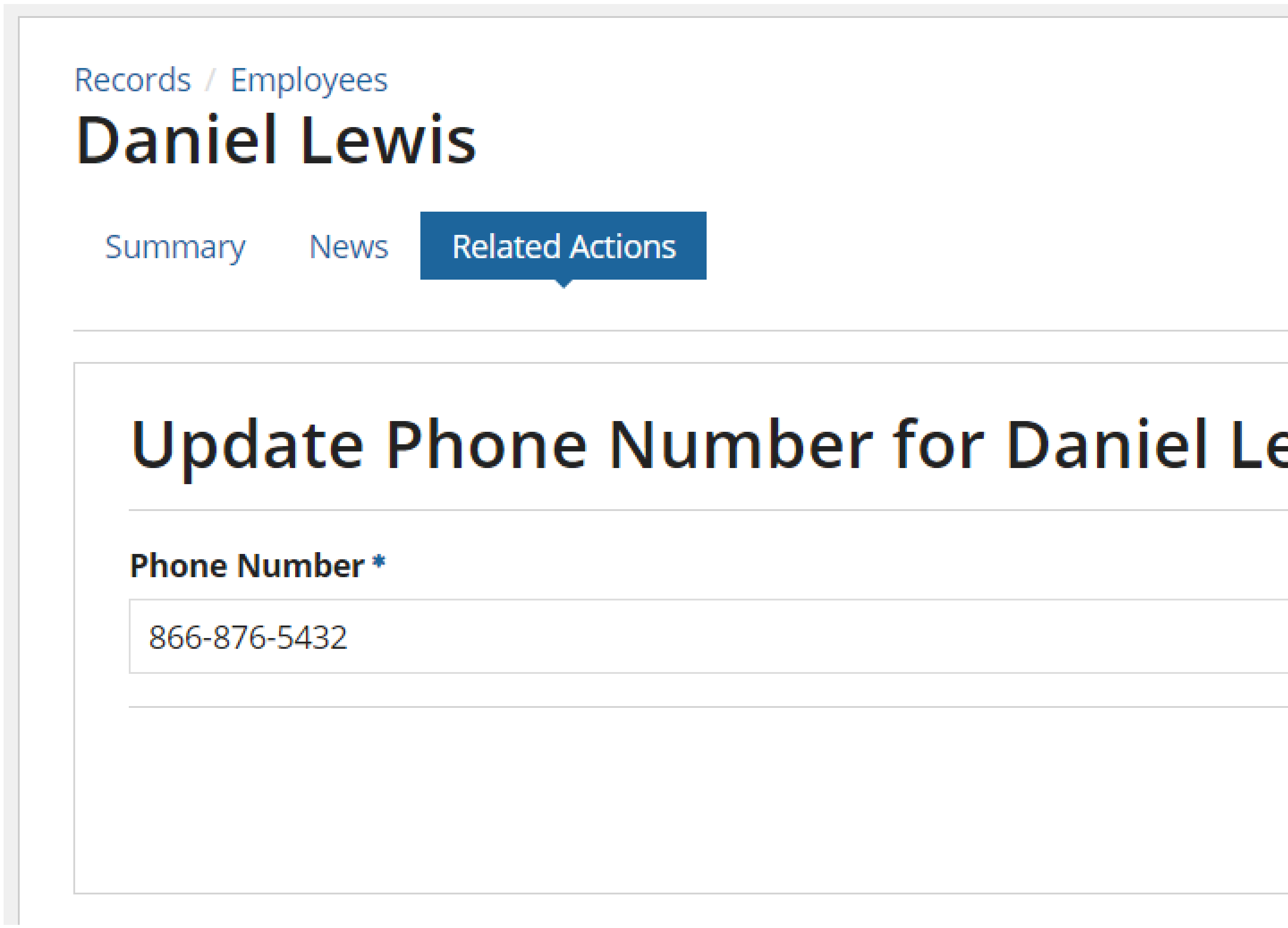
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Records / Employees

Daniel Lewis

Summary News Related Actions

Update Phone Number for Daniel Le

Phone Number *

866-876-5432

Enter a new number, click **Submit**, and refresh Daniel Lewis' record. You should see the updated number.

In addition to the Related Actions view, you can configure specific related actions to be available from specific record views. Let's try this by making the Update Phone Number related action available from the Summary view.

1. In the Views section, click on the Summary view to edit it.
2. In the Related Actions Shortcut field, check the Update Phone Number related action and click **OK**

Use the Report Builder

Overview

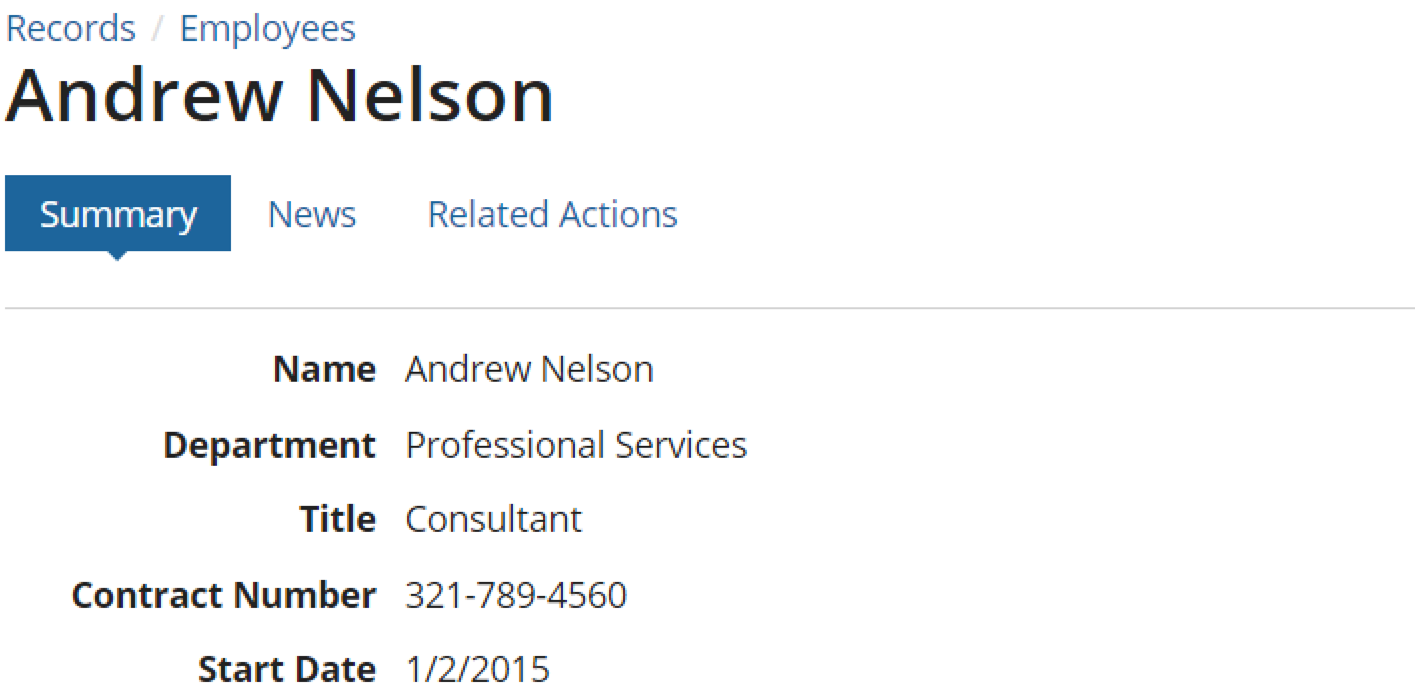
This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

3. Click **Save** in the header of the Record Type Designer.

Refresh the Employees record in Tempo and select an employee from the list. From the Summary view, you should see that the Update Phone Number related action is available as a button on the top right.



Name	Andrew Nelson
Department	Professional Services
Title	Consultant
Contract Number	321-789-4560
Start Date	1/2/2015

For more information about configuring related actions, see Add a Related Action.

Export Record List to Excel

Designers can allow record viewers to export record lists to Excel. This setting displays an *Export to Excel* button on a record list that record viewers can click to download a copy of their filtered record lists in Excel.

To enable this option, check the checkbox next to **Show Export to Excel Button** in the Record List section of your Record Type object.

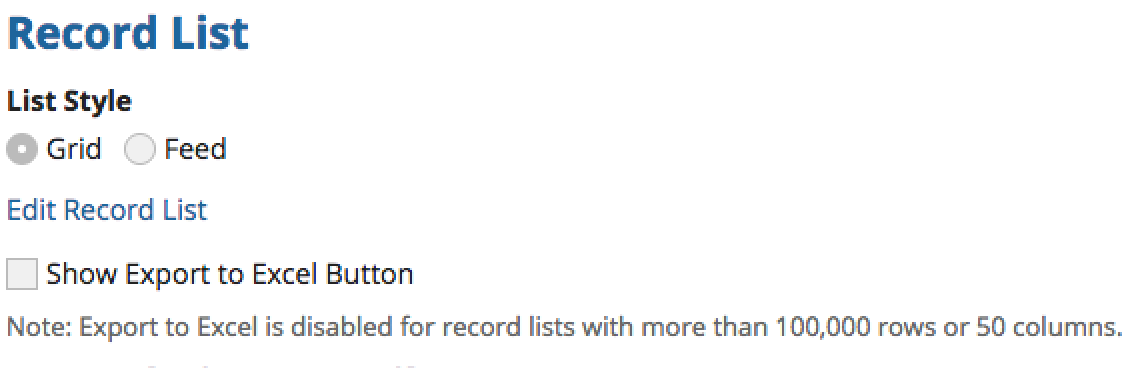
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Record List

List Style

● Grid ○ Feed

Edit Record List

☐ Show Export to Excel Button

Note: Export to Excel is disabled for record lists with more than 100,000 rows or 50 columns.

Note: This functionality is only available for data store entity backed record lists using a grid list style. The button will be disabled if the filtered record list contains more than 100,000 rows or 50 columns.

Give Users Access to View the Records

Since we didn't add any users or groups to the record type security, only system administrators can view the records in Tempo. By adding groups and giving them Viewer, Editor, or Administrator permissions, you can open up the records to other users.

1. Open the Employee Directory record type in the record type designer.
2. Open the **Settings** menu and click on **Security**.
3. Click the link to add **Users or Groups**.
4. Pick the groups containing the users to whom you want the record to be visible. Set their permission level to **Viewer**
5. Click **Add** and then **Save Changes**.

All members of the specified groups can now see the Employee record. You can repeat these steps for the other roles.

For more information on security for record types, see Configuring Security for a Record Type.

Create Process-Backed Records

Now that we've created an entity-backed record, let's move on to process-backed records.

Before we start, though, we'll need a process model. For our example, let's use the process model from the Process Model Tutorial. If you haven't already created the Submit Expenses process model, do so now by completing the first five sections of the tutorial, then follow the steps below to create your record type.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

1. Navigate to the application contents view of the **Appian Tutorial** application (if needed).
2. Click **New**, and then click **Record Type**.
3. In the **Create Record Type** dialog, enter the following information:
    - Name: Expense Report
    - Plural Name: Expense Reports
    - Description: List of expense reports
4. Click **Create & Edit**.

After the Expense Report record type opens in the record type designer, we need to add some more information:

1. Enter the following values into the Data section:
    - Source: Process Model
    - Process Model: Submit Expenses
2. In the Record List section, select **Feed** in the **List Style** field.
3. Click on the **Edit Record List** link and enter the following:
    - List View Item Template:

```
=a!listViewItem(
 title: rf!expenseItem
)
```

    - Sort Field: expenseDate
    - Sort Order: Descending
4. Click **OK** to close the dialog and save the record list settings.
5. Click on the Summary view in the Views grid and enter the following into the Interface field:

```
={
    a!textField(
      label: "Requester",
      labelPosition: "ADJACENT",
      value: user(touser(rf!pp.initiator), "firstName") & " " & user(touser(rf!pp.initiator),
"lastName"),
      readOnly: true
    ),
    a!textField(
      label: "Amount",
      labelPosition: "ADJACENT",
      value: rf!expenseAmount,
      readOnly: true
    ),
    a!textField(
      label: "Date",
      labelPosition: "ADJACENT",
      value: rf!expenseDate,
      readOnly: true
    )
 }
```

    - Your form should look like the following:

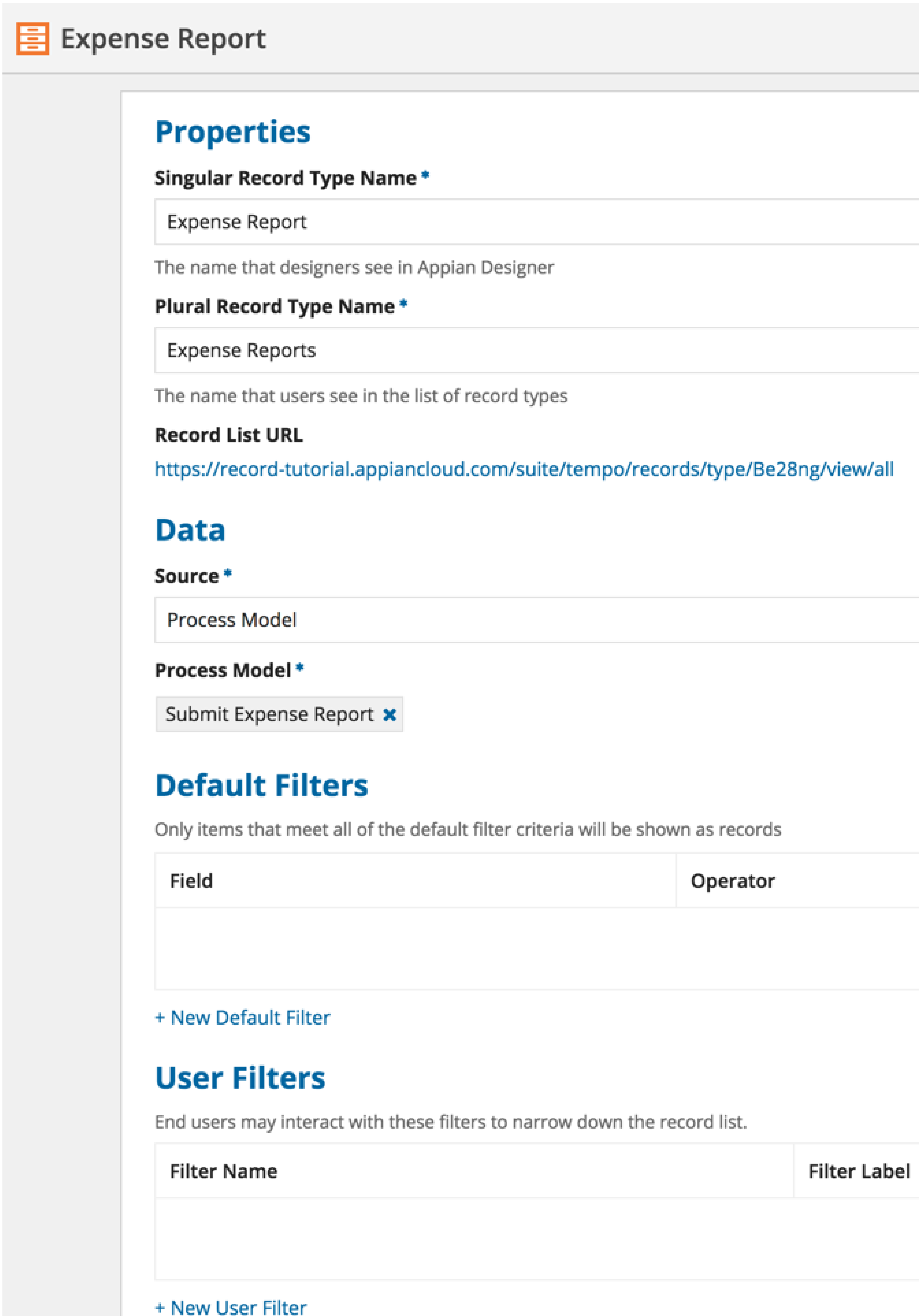
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Expense Report

Properties

Singular Record Type Name *

Expense Report

The name that designers see in Appian Designer

Plural Record Type Name *

Expense Reports

The name that users see in the list of record types

Record List URL

https://record-tutorial.appiancloud.com/suite/tempo/records/type/Be28ng/view/all

Data

Source *

Process Model

Process Model *

Submit Expense Report ✖

Default Filters

Only items that meet all of the default filter criteria will be shown as records

Field	Operator

+ New Default Filter

User Filters

End users may interact with these filters to narrow down the record list.

Filter Name	Filter Label

+ New User Filter

Us

Ove

This top

After fir
get star

The rep
type or
the repo

6. Click **Save** in the header of the Record Type Designer.

You have just created a record for each of the processes of the Submit Expenses process model.

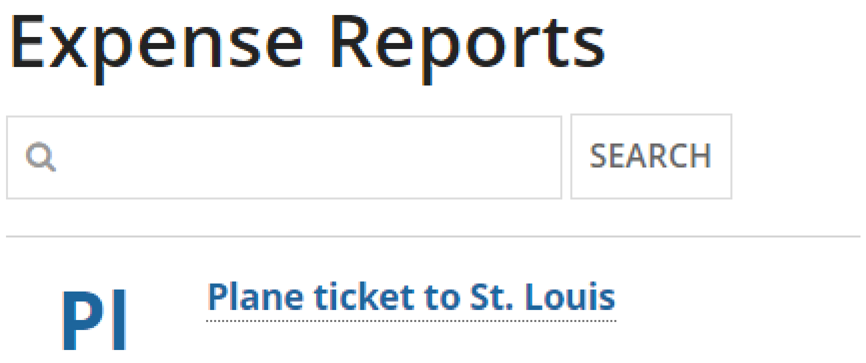
If you go the record list view in Tempo, however, you might not see any expense reports listed. This is because you don't have any instances of the process. Let's add one and take a look at the result.

Start an instance of the Submit Expenses process with the following values:

- Expense Item: Plane ticket to St. Louis
- Expense Amount: 200
- Expense Date: 4/16/2013
- Comments: Flights are expensive.

In Tempo, select the **Records** tab, and click **Expense Reports**.

You should see the following:



Select the "Plane ticket to St. Louis" record.

You should see the following:

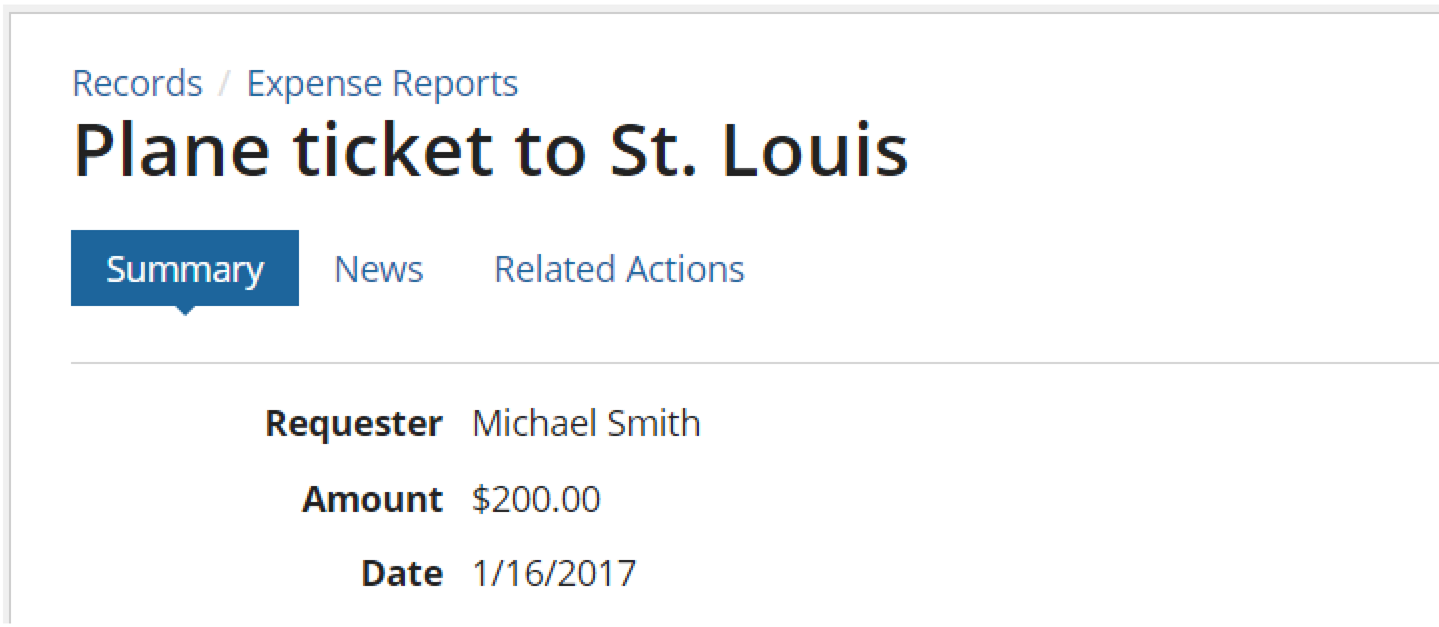
Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.


Records / Expense Reports

Plane ticket to St. Louis

Requester Michael Smith

Amount $200.00

Date 1/16/2017

Just like the entity-backed records, the design of the record view is based on the expression we added to the Interface field of the Summary view of the record type. You'll notice we used the `rf!` domain again, but this time it was to access process variables of the process model. This is required for process-backed records. Along with the process variables, you can also access certain process and process model properties using the same domain.

Only certain process and process model properties are available for use in interfaces.

For the full list, see also: Record List

Other than that, you can change this record view design as you would any other interface.

Similarity to Entity-backed Records

Notice that both the record list and summary view for process-backed records look the same as for entity-backed records. The same advanced configurations you did in the entity-backed record example are also applicable to process-backed records. The process for implementing these configurations is exactly the same as well. See the sections in the entity-backed records example for the step-by-step process.

Add Related Actions for Quick Tasks

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.

The only new concept for process-backed records that we did not cover with entity-backed records is exposing quick tasks as related actions. This is because entities can't have quick tasks; only processes can. However, exposing your processes quick tasks as related actions on the record is easy.

If the process model you used as the source for the record had any quick tasks (the example we used did not), the quick tasks that are available to the user would automatically appear in the list of related actions for those records. You don't have to take any further action to get them to appear in the list. However, you cannot make quick tasks available to end users as related actions from record views. You are also able to configure other related actions following the same example we saw for entity-backed records.

Use the Report Builder

Overview

This topic walks you through creating a report with the report builder.

After first creating an interface we provide a list of template, examples, and builders that will help you get started in design mode.

The report builder allows you to create basic grids and charts to display data from a record type or data store entity using an intuitive interface. This page walks through the main components of the report builder.