

**Appian**

# Task Report Tutorial

Appian 7.7

The walkthrough on this page will help you create your first task report as a SAIL interface.

Toggle Us

Task reports are intended to display task information with a link for users to open each task and begin working on it. When you save a Tempo report as a task report, it appears on the Tasks tab of the Work Platform environment.

To create a task report, use the GridField component. This way you'll get a nice grid layout of the information and you can add a column for the task links. To retrieve the task information, use the Portal to SAIL function plug-in available on Forum. This plug-in helps you query existing Portal task reports and populate the GridField component with this information.

For our example, we'll pull data from the My Tasks Portal report. We'll describe how to use the Portal to SAIL plug-in to display data from the Portal report in a GridField component. Then, we'll show you how to create links to tasks and format the data so it looks more user-friendly. Finally, we'll show you how to add dynamic filters to your report.

Use the data provided to understand how the configurations work. Then, try it with your own data.

1. [Create a Task Report](#)
2. [Display the Task Data in a Grid](#)
3. [Create Links to Process Tasks](#)
4. [Format the Status Column](#)
5. [Add a Predefined Filter](#)

The content below assumes a basic familiarity with SAIL interfaces, specifically the GridField and DropdownField components, and focuses more on the specifics of designing and creating a task report. Consider going through the SAIL and Grid Tutorials first and taking a look at the Tempo Report Design page before proceeding.

See also: [Tempo Report Design](#), [SAIL Tutorial](#), and [Grid Tutorial](#)

## Create a Task Report


First, let's show you how to create a task report and access it from the Tempo interface.

Create a group called " **My Tasks Report Admin** " that includes only your username. This group defines who has administrator access to the Tempo report.

See also: [Creating Groups](#)

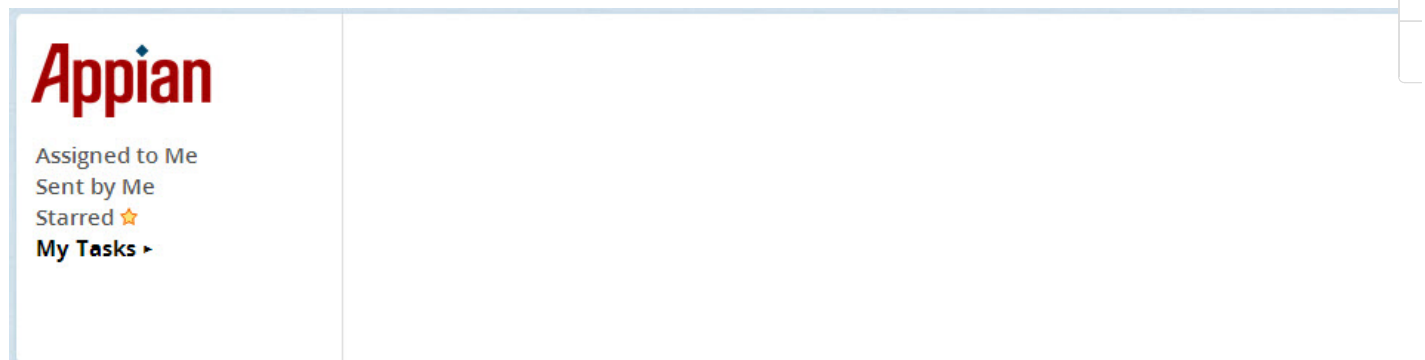
From the left navigation of the Reports tab in the Designer interface, click **Create a Tempo Report**.

Enter the following values for the associated fields:

- Name: My Tasks
- Administrator Group: "My Tasks Report Admin"
- Save as Task Report: checked
- SAIL Dashboard: 

Click **Create Report**.

To view your report, open Tempo in a separate window and select the Tasks tab. You'll now see a link for the **My Tasks** report below the default filters as shown below:



The task report is currently blank since we haven't created the interface for it yet. Let's do that now.

## Display the Task Data in a Grid

Next, let's use the Portal to SAIL plug-in to populate a GridField component with data from the My Tasks Portal report and create a Tempo task report out of it.

Download the Portal to SAIL plug-in available in the Shared Components record type on Forum and deploy it to your environment.

See also: [Portal to SAIL Plug-In](#)

This plug-in includes two functions. We'll be using the `getPortalReportDataSubset()` function. It works in a similar manner to a query rule or the result of the `toDataSubset()` function.

The record dashboard for this plug-in includes an instructions attachment where you can find important information on the functionality and syntax of this function. Take a look at this page before continuing.

See also: [Appian Plug-ins](#)

Next, create a constant called `MY_TASKS_REPORT` of type Document with the "My Tasks" report as the value. This report is located in the Communities > Default Community > System Knowledge Center > System Reports > Task Reports folder.

See also: [Constants](#)

From the Rules view in the Designer interface, create an expression rule called `getMyTasks` with the following definition:

```
=getPortalReportDataSubset(
  reportId: cons!MY_TASKS_REPORT,
  pagingInfo: ri!pagingInfo
)
```

Add the following input to this rule:

- pagingInfo (Any Type)

See also: [Rules](#)

To test the rule, create another expression rule called `testGetMyTasks` with the following definition:

```
=rule!getMyTasks(pagingInfo: topaginginfo(1, -1))
```

Click **Test Rule**.

The rule output will render the text representation of a DataSubset value similar to the following:

```
[startIndex=1,
 batchSize=-1,
 sort=,
 totalCount=2,
 data=
  [cells=[value=Single Group Assigned, drilldown=268437065]; [value=Multiple Group Assigned, drilldown=268437067]];
  [cells=[value=9/27/2013 2:56 PM GMT+00:00, drilldown=]; [value=9/27/2013 2:56 PM GMT+00:00, drilldown=]];
  [cells=[value=1, drilldown=]; [value=1, drilldown=]];
  [cells=[value=group assigned tasks, drilldown=268435514]; [value=group assigned tasks, drilldown=268435514]];
  [cells=[value=0, drilldown=268435514]; [value=0, drilldown=268435514]];
  [cells=[value=, drilldown=]; [value=, drilldown=]];
  [cells=[value=[Group:6], drilldown=]; [value=[Group:6]; [Group:7]; [Group:3], drilldown=]],
 identifiers=268437065; 268437067,
 report=[name=My Tasks, description=A list of all tasks for the current user.],
 columns=
  [label=Name, field=0, alignment=LEFT];
  [label=Received, field=2, alignment=LEFT];
  [label=Priority, field=4, alignment=LEFT];
  [label=Process, field=3, alignment=LEFT];
  [label=Status, field=5, alignment=LEFT];
  [label=Deadline, field=7, alignment=LEFT];
  [label=Assigned To, field=8, alignment=LEFT]
]
```

Notice how the data for the report columns is returned in the `data` field, while the remaining column configurations are in the `columns` field. These arrays will be used to configure each column.

We'll use the index of each column in the arrays when we go to format the data for each column, and we'll use the field number of the columns when we go to configure a filter and default sorting.

This rule was only used for testing your expression, so you don't need to save it.

Now let's create a grid that displays a few of the columns from the My Tasks report. For this example, we'll take the columns labeled `Name`, `Process`, and `Status`.

Let's create this interface using the Interface Designer. From the Rules tab of the Designer interface, click **Create an Interface**. Here you can quickly create and modify your interface with the ability to immediately see and test the result.

See also: [Creating an Interface](#)

Enter the following in the design pane on the left-hand side:

```
=load(
  local!pagingInfo: a!pagingInfo(startIndex: 1, batchSize: 5),
  with(
```

```

local!datasubset: rule!getMyTasks(pagingInfo: local!pagingInfo),
a!gridField(
  label: local!datasubset.report.name,
  instructions: local!datasubset.report.description,
  columns: {
    a!gridTextColumn(
      label: local!datasubset.columns[1].label,
      field: local!datasubset.columns[1].field,
      data: if(
        local!datasubset.totalCount=0,
        {},
        local!datasubset.data[1].cells.value
      ),
      alignment: local!datasubset.columns[1].alignment
    ),
    a!gridTextColumn(
      label: local!datasubset.columns[4].label,
      field: local!datasubset.columns[4].field,
      data: if(
        local!datasubset.totalCount=0,
        {},
        local!datasubset.data[4].cells.value
      ),
      alignment: local!datasubset.columns[4].alignment
    ),
    a!gridTextColumn(
      label: local!datasubset.columns[5].label,
      field: local!datasubset.columns[5].field,
      data: if(
        local!datasubset.totalCount=0,
        {},
        local!datasubset.data[5].cells.value
      ),
      alignment: local!datasubset.columns[5].alignment
    )
  },
  value: local!pagingInfo,
  saveInto: local!pagingInfo,
  totalCount: local!datasubset.totalCount
)
)
)
)

```

You should see an interface similar to the following (it will vary somewhat based on your personal task list):

**My Tasks**  
 A list of all tasks for the current user.

Name	Process	Status
Review Purchase Request: Purchase Request Number 68	New PR: Purchase Request Number 68	0
Approve Purchase Request: Purchase Request Number 43	New PR: Purchase Request Number 43	1
Approve Purchase Request: Purchase Request Number 98	New PR: Purchase Request Number 98	0

If you've already gone through the Grid Tutorial, you'll notice how we used a local variable to create the data and used the indices of the **Name**, **Process**, and **Status** columns ( **[1]**, **[4]**, and **[5]** respectively) to populate the columns of our grid.

You can add more columns by adding another item to the columns argument that calls the **a!gridTextColumn()** function but uses a different column index number.

Now let's save the interface with the name **myTasksReport**. Keep this window open so you can quickly modify the interface as we continue through the following sections.

## Create Links to Process Tasks

Unlike the Portal report, the grid we just created only displays the task name. Users have no way to open the task. We need to configure this separately.

For our example, let's change the **Name** column of the grid to link to the process tasks.

Modify the **myTasksReport** interface with the following (the grey text just indicates what was a part of the previous expression so you can easily see what we added):

```
=load(
  local!pagingInfo: a!pagingInfo(startIndex: 1, batchSize: 5),
  with(
    local!datasubset: rule!getMyTasks(pagingInfo: local!pagingInfo),
    a!gridField(
      label: local!datasubset.report.name,
      instructions: local!datasubset.report.description,
      columns: {
        a!gridTextColumn(
          label: local!datasubset.columns[1].label,
          field: local!datasubset.columns[1].field,
          data: if(
            local!datasubset.totalCount=0,
            {},
            local!datasubset.data[1].cells.value
          ),
          links: apply(
            a!processTaskLink(label: _, task: _),
            merge(
              local!datasubset.data[1].cells.value,
              local!datasubset.data[1].cells.drilldown
            )
          ),
          alignment: local!datasubset.columns[1].alignment
        ),
        a!gridTextColumn(
          label: local!datasubset.columns[4].label,
          field: local!datasubset.columns[4].field,
          data: if(
            local!datasubset.totalCount=0,
            {},
            local!datasubset.data[4].cells.value
          ),
          alignment: local!datasubset.columns[4].alignment
        ),
        a!gridTextColumn(
          label: local!datasubset.columns[5].label,
          field: local!datasubset.columns[5].field,
          data: if(
            local!datasubset.totalCount=0,
            {},
            local!datasubset.data[5].cells.value
          ),
          alignment: local!datasubset.columns[5].alignment
        )
      },
    value: local!pagingInfo,
    saveInto: local!pagingInfo,
    totalCount: local!datasubset.totalCount
  )
)
```

You should see the following:

**My Tasks**

A list of all tasks for the current user.

Name	Process	Status
<a href="#">Review Purchase Request: Purchase Request Number 68</a>	New PR: Purchase Request Number 68	0
<a href="#">Approve Purchase Request: Purchase Request Number 43</a>	New PR: Purchase Request Number 43	1
<a href="#">Approve Purchase Request: Purchase Request Number 98</a>	New PR: Purchase Request Number 98	0

This expression uses the `a!processTaskLink()` function to create a link for each data point. It passes the Portal report column's `value` (in this case, the task name) to the `label` parameter, which is used for the link's tooltip, and the column's `drilldown` (in this case, the task id) to the `task` parameter. We used it within the `apply()` function to create a link for each data point that is returned since this number can vary.

See also: [Looping Functions](#) and [Process Task Link](#)

## Format the Status Column

You may have noticed that the `Status` column returned numbers instead of the text you normally see in the Portal task report. This is because the Portal report auto-formats some columns, including task status, task priority, users, dates, and more. The `getPortalReportDataSubset()` function, however, does not auto-format these values.

We can format them ourselves by adding a rule for each new format and modifying our task report rule to use them.

Create an expression rule called `getTaskStatusDisplay` with the following definition:

```
=displayvalue(
  ri!status,
  enumerate(13),
  {"Assigned", "Accepted", "Completed", "Not Started", "Cancelled", "Paused", "Unattended", "Aborted", "Cancelled By Exception", "Submitted", "Running", "Error", "Skipped"},
  ""
)
```

Add the following input to this rule:

- status (Integer)

This rule assigns a text value to each number returned by the function.

Modify the `myTasksReport` interface with the following:

```
=load(
  local!pagingInfo: a!PagingInfo(startIndex: 1, batchSize: 5),
  with(
    local!datasubset: rule!getMyTasks(pagingInfo: local!pagingInfo),
    a!gridField(
      label: local!datasubset.report.name,
      instructions: local!datasubset.report.description,
      columns: {
        a!gridTextColumn(
          label: local!datasubset.columns[1].label,
          field: local!datasubset.columns[1].field,
          data: if(
            local!datasubset.totalCount=0,
            {},
            local!datasubset.data[1].cells.value
          ),
          links: apply(
            a!processTaskLink(label: _, task: _),
            merge(
              local!datasubset.data[1].cells.value,
              local!datasubset.data[1].cells.drilldown
            )
          ),
          alignment: local!datasubset.columns[1].alignment
        ),
        a!gridTextColumn(
          label: local!datasubset.columns[4].label,
```

```

    field: local!datasubset.columns[4].field,
    data: if(
      local!datasubset.totalCount=0,
      {},
      local!datasubset.data[4].cells.value
    ),
    alignment: local!datasubset.columns[4].alignment
  ),
  a!gridTextColumn(
    label: local!datasubset.columns[5].label,
    field: local!datasubset.columns[5].field,
    data: if(
      local!datasubset.totalCount=0,
      {},
      apply(
        rule!getTaskStatusDisplay,
        local!datasubset.data[5].cells.value
      )
    ),
    alignment: local!datasubset.columns[5].alignment
  )
},
value: local!pagingInfo,
saveInto: local!pagingInfo,
totalCount: local!datasubset.totalCount
)
)
)
)

```

You should see an interface similar to the following:

**My Tasks**

A list of all tasks for the current user.

Name	Process	Status
<a href="#">Review Purchase Request: Purchase Request Number 68</a>	New PR: Purchase Request Number 68	Assigned
<a href="#">Approve Purchase Request: Purchase Request Number 43</a>	New PR: Purchase Request Number 43	Accepted
<a href="#">Approve Purchase Request: Purchase Request Number 98</a>	New PR: Purchase Request Number 98	Assigned

## Add a Predefined Filter

With Portal reports, Appian lets users add their own filters. For task reports, we can use our `getPortalReportDatasubset()` plug-in to define them. Then we can connect them to another SAIL component so users can use that component to change the filter on the report.

For our example, let's add a DropdownField component so users can filter the tasks by the most common statuses.

Open your `getMyTasks` expression rule and modify the definition with the following:

```

=getPortalReportDatasubset(
  reportId: cons!MY_TASKS_REPORT,
  pagingInfo: ri!pagingInfo,
  contextIds: {},
  filter: ri!filters
)

```

Add the following input to this rule:

- filters (Any Type)

We only really need to use the `filter` field, but since this is a function plug-in and not a system-delivered SAIL function, we need to pass inputs in the order expected by the function.

Modify the `myTasksReport` interface with the following:

```

=load(
  local!pagingInfo: a!PagingInfo(startIndex: 1, batchSize: 5),

```

```

local!statusFilter: 1,
with(
  local!datasubset: rule!getMyTasks(
    pagingInfo: local!pagingInfo,
    filters: if(
      isnull(local!statusFilter),
      {},
      {field: 5, operator: "EQUAL", value: local!statusFilter}
    )
  ),
a!dashboardLayout(
  firstColumnContents: {
    a!dropdownField(
      label: "Status",
      choiceLabels: {"Assigned", "Accepted", "Completed", "Not Started"},
      choiceValues: enumerate(4),
      placeholderLabel: "All",
      value: local!statusFilter,
      saveInto: local!statusFilter
    ),
    a!gridField(
      label: local!datasubset.report.name,
      instructions: local!datasubset.report.description,
      columns: {
        a!gridTextColumn(
          label: local!datasubset.columns[1].label,
          field: local!datasubset.columns[1].field,
          data: if(
            local!datasubset.totalCount=0,
            {},
            local!datasubset.data[1].cells.value
          ),
          links: apply(
            a!processTaskLink(label: _, task: _),
            merge(
              local!datasubset.data[1].cells.value,
              local!datasubset.data[1].cells.drilldown
            )
          ),
          alignment: local!datasubset.columns[1].alignment
        ),
        a!gridTextColumn(
          label: local!datasubset.columns[4].label,
          field: local!datasubset.columns[4].field,
          data: if(
            local!datasubset.totalCount=0,
            {},
            local!datasubset.data[4].cells.value
          ),
          alignment: local!datasubset.columns[4].alignment
        ),
        a!gridTextColumn(
          label: local!datasubset.columns[5].label,
          field: local!datasubset.columns[5].field,
          data: if(
            local!datasubset.totalCount=0,
            {},
            apply(
              rule!getTaskStatusDisplay,
              local!datasubset.data[5].cells.value
            )
          ),
          alignment: local!datasubset.columns[5].alignment
        )
      },
      value: local!pagingInfo,
      saveInto: local!pagingInfo,
      totalCount: local!datasubset.totalCount
    )
  )
)
)
)
)

```

You should see an interface similar to the following:

**Status**  
Accepted ▼

**My Tasks**  
A list of all tasks for the current user.

Name	Process	Status
<a href="#">Approve Purchase Request: Purchase Request Number 43</a>	New PR: Purchase Request Number 43	Accepted

In this expression, we set the *choiceLabels* to the task statuses and *choiceValues* to the numeric values that correspond to those statuses. Because the numeric values of these specific statuses are 0, 1, 2, and 3, we can use the *enumerate* function to return those values. Since we set the default value of the filter to 1 which corresponds to the first index of the *choicesLabels* array, only the tasks with a status of *Assigned* are returned originally. you select a different value from the dropdown, the grid updates to display only tasks that correspond to the selected status. If you select the placeholder, *local!statusFilter* is set to null and we do not pass a filter value to ensure that all tasks are returned.

You may have seen an error if you were on any page other than the first page of the grid when you selected a new filter value. This is because the start index of the paging configuration persisted across reevaluations may not apply when you select a new value. If you are on the second page of the grid (for example when *local!pagingInfo.startIndex* is 6) when you change the filter, the start index that will be applied might be greater than the total number of tasks that can be returned for that filter, which causes an error. To avoid this, we need to reset the start index each time a new filter is selected.

Create an expression rule called *updatePagingInfo* with the following definition:

```
=a!pagingInfo(
  startIndex: 1,
  batchSize: ri!pagingInfo.batchSize,
  sort: ri!pagingInfo.sort
)
```

Add the following inputs to this rule: - *pagingInfo* (Any Type) This rule will take in a *PagingInfo* and reset the start index to `1` while preserving the other fields. Modify the *myTasksReport* interface with the following:

```
=load(
  local!pagingInfo: a!PagingInfo(startIndex: 1, batchSize: 5),
  local!statusFilter: 1,
  with(
    local!datasubset: rule!getMyTasks(
      pagingInfo: local!pagingInfo,
      filters: if(
        isnull(local!statusFilter),
        {},
        {field: 5, operator: "EQUAL", value: local!statusFilter-1}
      )
    ),
    a!dashboardLayout(
      firstColumnContents: {
        a!dropdownFieldByIndex(
          label: "Status",
          choiceLabels: {"Assigned", "Accepted", "Completed", "Not Started", "Cancelled", "Paused", "Unattended", "Aborted", "Cancelled By Exception", "Submitted", "Running", "Error", "Skipped"},
          placeholderLabel: "All",
          value: local!statusFilter,
          saveInto: {
            local!statusFilter,
            a!save(local!pagingInfo, rule!updatePagingInfo(local!pagingInfo))
          }
        ),
      },
      a!gridField(
        label: local!datasubset.report.name,
        instructions: local!datasubset.report.description,
        columns: {
          a!gridTextColumn(
            label: local!datasubset.columns[1].label,
            field: local!datasubset.columns[1].field,
            data: if(
              local!datasubset.totalCount=0,
```



```


    {},
    local!datasubset.data[1].cells.value
  ),
  links: if(
    local!datasubset.totalCount=0,
    {},
    apply(
      a!processTaskLink(label: _, task: _),
      merge(
        local!datasubset.data[1].cells.value,
        local!datasubset.data[1].cells.drilldown
      )
    )
  ),
  alignment: local!datasubset.columns[1].alignment
),
a!gridTextColumn(
  label: local!datasubset.columns[4].label,
  field: local!datasubset.columns[4].field,
  data: if(
    local!datasubset.totalCount=0,
    {},
    local!datasubset.data[4].cells.value
  ),
  alignment: local!datasubset.columns[4].alignment
),
a!gridTextColumn(
  label: local!datasubset.columns[5].label,
  field: local!datasubset.columns[5].field,
  data: if(
    local!datasubset.totalCount=0,
    {},
    apply(
      rule!getTaskStatusDisplay,
      local!datasubset.data[5].cells.value
    )
  ),
  alignment: local!datasubset.columns[5].alignment
),
},
value: local!pagingInfo,
saveInto: local!pagingInfo,
totalCount: local!datasubset.totalCount
)
}
)
)
)
)

```

You should see the same interface as you saw earlier.

Now let's add this interface to our task report. In the Reports tab of the Designer interface, click on **My Tasks Report**. In the SAIL Dashboard field, enter the following: `rule!myTasksReport()`

Click **Save Report**. In the Tempo environment, go to the Tasks tab and click on **My Tasks**. You should see the following:



Assigned to Me  
Sent by Me  
Starred ★  
**My Tasks ▶**

**Status**

Accepted ▼

**My Tasks**

A list of all tasks for the current user.

Name	Process	Status
<a href="#">Approve Purchase Request: Purchase Request Number 43</a>	New PR: Purchase Request Number 43	Accepted

